Analysis of TTL-based Cache Networks

N. Choungmo Fofack
cn916667@etu.unice.fr

Philippe Nain
philippe.nain@inria.fr

Giovanni Neglia
giovanni.neglia@inria.fr

Don Towsley
towsley@cs.umass.edu

ABSTRACT

Many researchers have been working on the performance analysis of caching in Information-Centric Networks (ICNs) under various replacement policies like Least Recently Used (LRU), FIFO or Random (RND). However, no exact results are provided, and many approximate models do not scale even for the simple network of two caches connected in tandem. In this paper, we introduce a Time-To-Live based policy (TTL), that assigns a timer to each content stored in the cache and redraws the timer each time the content is requested (at each hit/miss). We show that our TTL policy is more general than LRU, FIFO or RND, since it is able to mimic their behavior under an appropriate choice of its parameters. Moreover, the analysis of networks of TTL-based caches appears simpler not only under the Independent Reference Model (IRM, on which many existing results rely) but also with the Renewal Model for requests. In particular, we determine exact formulas for the performance metrics of interest for a linear network and a tree network with one root cache and N leaf caches. For more general networks, we propose an approximate solution with the relative errors smaller than 10^{-3} and 10^{-2} for exponentially distributed and constant TTLs respectively.

1. INTRODUCTION

Caches are widely used in networks and distributed systems for improving performance. They are integral components of the Web [5], DNS [17], and Content Distribution Networks (CDNs) [25]. More recently there has been a growing emphasis on *content networks* (i.e. CCNs) where content or data is centric and host-to-content interaction is the common case [24]. Many of these systems give rise to hierarchical (tree) cache topologies and to even more general irregular topologies. The design, configuration, and analysis of these cache systems pose significant challenges. An abundant literature exists on the performance (e.g. hit probability, search cost) of a single cache running the LRU replacement policy or, its companion, the Move-to-Front (MTF) policy (see [3, 18, 11, 2, 1, 10, 8, 12, 13] for i.i.d. requests and [7, 15, 14] for correlated requests). With few exceptions, exact models of even single caches are computationally intractable, resulting in the reliance on approximations [8, 13]. Networks of caches are significantly more difficult to analyze and no exact solution has been obtained so far for even the simple configuration of two LRU caches in series. A few approximations have been proposed, instead, for a simple two-level LRU cache network [5] and a general LRU network [23]. However, their inaccuracies can be significant as reported in [23] where the relative error reaches 16%.

In this paper, we focus on a class of caches referred to as Time-To-Live (TTL) caches. When an uncached data is brought back into the cache due to a cache miss, a local TTL is set. The TTL value can be different for different data, but also for the same data at different caches[1]. All requests to that data before the expiration of the TTL are successful (cache hit); the first request for that data to arrive after the TTL expiration will yield a cache miss. In that case, the cache may forward the request to a higher-level cache, if any, or to the server. When located, the data is routed on the reverse-path and a copy is placed in each cache along the path. We strongly support the idea that a TTL policy can be an interesting alternative to more popular schemes like LRU or FIFO for two reasons. First, a TTL policy is more configurable and in particular can mimic the behavior of other replacement policies for an opportune choice of its parameters (the timeout values, see Section 3.2). Second, networks of TTL caches appear to be simpler to study analytically, while networks of LRU or FIFO caches have defeated until now modeling attempts. As we shall show, our TTL policy is general enough and fully configurable, and it appears to be a suitable unified framework for the performance analysis of heterogeneous caching networks where caches may run under different replacement policies.

We develop in particular a set of building blocks for the performance evaluation of hierarchical TTL cache networks with general topologies and where TTLs are set with every request. These blocks allow one to model exogenous requests at different caches as independent renewal processes and to allow for TTL duration to be described by an arbitrary distribution as long as they are independent of each other. The building blocks consist of:

[1]This is then different from the TTLs in DNS system where the TTL for a given data is in general set to a common value determined by the authoritative name server.

- a renewal theoretic model of a *single content* TTL cache when fed by a renewal request stream,
- a renewal process approximation of the superposition of independent renewal processes.

The first block forms the basis for calculating cache metrics such as miss and hit probabilities/rates but also for characterizing the output sequence of requests (the miss process) of a cache, while the second block is used to characterize the resulting process of the superposition of exogenous requests and those resulting from a miss process if any. We apply these blocks to the case when TTL values are either exponentially distributed or constant. Then, we focus primarily on linear TTL networks, two level TTL tree networks and combinations of the two. We derive exact results for some cases but when they are not, the relative errors are extremely small and less than 10^{-3} in the case of exponentially distributed TTLs and 10^{-2} in the case of constant TTLs. Thus, we believe our approach is promising and capable of accurately modeling a richer class of network topologies.

In the literature, the paper closer to our approach is [16], where the authors consider a *single* TTL-based cache fed by i.i.d. requests to a single data. They obtain the hit rate for a constant TTL via the solution of some renewal equation. Despite the increasing interest in CCNs, previous work has mainly focused on global architecture design. [4] is probably the first attempt to model data transfer in CCNs; the authors develop approximations to calculate the stationary throughput in a network of LRU caches taking into account the interplay between receiver driven transport and per-chunk caching.

The paper is organized as follows. In Section 2, we introduce notation, the model assumptions, and a key result from [22] regarding the computation of the marginal inter-arrival distribution for a superposition of arrival streams modeled as renewal processes. Section 3 contains our renewal theoretic model along with its application to some network topologies where it leads to exact results. It also shows how the TTL policy can mimic existing policies. We describe in section 4 our approach to model the combined exogenous and miss streams as a renewal request process; and the resulting approximations for a larger class of linear and tree networks. We also report the accuracy of this approximation. Section 7 shows how the TTL policy behaves under finite size buffer. Conclusions are found in Section 8.

Due to space constraints, all the proofs have been omitted as well as a discussion of the computational complexity of our analytic approach, but they can be found in the companion technical report [6] together with more examples and a more detailed validation section.

2. DEFINITIONS AND ASSUMPTIONS

Throughout this paper we mainly focus on particular instances of TTL cache tree networks with infinite buffers size. This key hypothesis allow us to decouple the management of the different contents and study each of them separately. For this reason, in what follows we will simply refer to a *single* content or data chunk (simply called the data). The effect of finite buffer is considered in Section 7. From now on

λ	Arrival rate (single cache)
$1/\mu$	Expected TTL (single cache)
$F(t)$	CDF exogenous arrivals (single cache)
$G(t)$	CDF inter-miss times (single cache)
$T(t)$	CDF TTL duration (single cache)
λ_n	Exogenous arrival rate at cache n
Λ_n	Overall arrival rate at cache n
$\Lambda_{n,k}$	Overall arrival rate at cache n for content k
$1/\mu_n$	Expected TTL at cache n
$F_n(t)$	CDF exogenous arrivals at cache n
$H_n(t)$	CDF overall arrivals at cache n
$H_{n,k}(t)$	CDF of overall arrivals at cache n for content k
$G_n(t)$	CDF inter-miss times at cache n
$T_n(t)$	CDF TTL duration at cache n
$h_{P,n}, m_{P,n}$	Hit, miss probability resp. at cache n
$h_{R,n}, m_{R,n}$	Hit, miss rate resp. at cache n
π_n	Occupancy of cache n (stationary probability content is in cache n
$\pi_{n,k}$	Occupancy of cache n for content k
q_n	Average size of cache n (= π_n if single content in network)
h_P, m_P	Hit, miss probability resp. (single cache)
h_R, m_R	Hit, miss rate resp. (single cache)
$\mathcal{C}(n)$	Set of children of cache n
$\chi^*(s)$	LST of CDF $\chi(t)$

Table 1: Glossary of main notation

the words "node" and "cache" will be used interchangeably. Also, a cache will always be a TTL cache unless otherwise specified.

New requests for a data can be generated at any node of the network according to mutually independent renewal processes; these requests are referred to as *exogenous* requests or arrivals and the sequence of these exogenous request instants is called the *exogenous request process.* If upon the arrival of a new request the data is not present in the cache, the request is *instantaneously* forwarded to the next level of the tree and the process repeats itself until the data is found. In case the data cannot be found along the path toward the root, the root retrieves it from a server. Once the data is found, either at a cache or at a server, a copy of it is *instantaneously* transmitted to each cache along the path between the cache where the data was found and the cache that issued the request[2]. A new TTL is set for each new copy of the data and the TTL is redrawn at the cache, if any, where the data was found (by convention, the TTL at the server is infinite). This is in contrast with the model in [16] where there is no TTL reset upon a cache hit. Resetting the TTL also at each cache hit increases the occupancy and the hit probability specially for popular contents (high request rate). This choice is motivated by the CCN paradigm of moving popular documents as close as possible to the users.

We define the *miss process* at a cache as the successive instants at which misses occur at this cache, namely, the times

[2]We observe that our TTL-based policy could be used also in conjunction with other strategies that do not keep a copy at every cache along a path, like those in [20].

at which the data is requested and is not found in the cache. Let us denote by $\mathcal{C}(n)$ the set of children of cache n. The (overall) *request process*, also called the *arrival process*, at cache n is the superposition of the miss processes of caches in $\mathcal{C}(n)$ and of the exogenous request process at cache n, if any. We assume that successive TTLs at each cache are i.i.d. random variables, that TTLs at different caches are mutually independent, and that all TTLs are independent of the exogenous arrivals. If Λ_n is the arrival rate of requests at cache n, $h_{P,n}$ (resp. $m_{P,n} = 1-h_{P,n}$) and $h_{R,n} = \Lambda_n h_{P,n}$ (resp. $m_{R,n} = \Lambda_n(1-h_{P,n})$) denote the stationary hit (resp. miss) probability and the stationary hit (resp. miss) rate at cache n, respectively. We denote by π_n the steady-state probability that the data is in cache n and we call it the *occupancy* of cache n. Hence, we just have to calculate $h_{P,n}$ and Λ_n.

For $t \geq 0$ and any non-negative random variable X with Cumulative Distribution Function (CDF) $\chi(t) = P(X < t)$, $\bar{\chi}(t) = 1 - \chi(t)$ is the Complementary Cumulative Distribution Function (CCDF), $\chi^\star(s) = E[e^{-sX}] = \int_0^\infty e^{-st} d\chi(t)$ $(s \geq 0)$ denotes its Laplace-Stieltjes Transform (LST), and $\hat{\chi}(s) = \int_0^\infty e^{-st}\chi(t)dt$ $(s > 0)$ denotes the Laplace Transform (LT) of $\chi(t)$.

The following result, taken from [22, Formula (4.1)], will be repeatedly used in this paper .

Theorem 2.1. *The CDF $R(t)$, of the inter-event times of the point process resulting from the superposition of K mutually independent renewal processes, is given by*

$$\bar{R}(t) \;=\; \sum_{k=1}^{K} \frac{\alpha_k}{\sum_{k=1}^{K}\alpha_k} \bar{R}_k(t) \prod_{j=1, j\neq k}^{K} \alpha_j \int_t^\infty \bar{R}_j(u)du,$$

with $R_k(t)$ and $\alpha_k > 0$, the CDF of the inter-event times and the arrival rate of process k, respectively.

3. EXACT RESULTS

3.1 Single cache

We consider a single TTL cache. Requests arrive at the cache according to a renewal process. Without loss of generality, we assume that the first request arrives at time $t = 0$ and finds an empty cache. We denote by X a generic inter-arrival time with CDF $F(t)$ and density $f(t)$. We also denote by T a generic TTL duration, with CDF $T(t)$. Since successive inter-arrival times and successive TTLs form two independent renewal sequences and since a miss triggers a new TTL, miss times are regeneration points of the state of the cache. This implies that inter-miss times form a renewal process, with generic inter-miss time denoted by Y and CDF $G(t)$. The stationary hit probability, hit rate and miss rate denoted by h_P, h_R and m_R, respectively, are given by

$$h_P = P(X \leq T) = \int_0^\infty F(t)dT(t), \tag{1}$$

$$h_R = \lambda h_P, \quad m_R = 1/E[Y] = \lambda(1 - h_P) \tag{2}$$

respectively, with $\lambda := 1/E[X]$ the arrival rate.

Propositions 3.1 and 3.2 (Proofs in [6]) respectively give the cache occupancy π and the CDF $G(t)$ for the most general result (with arbitrary CDFs $F(t)$ and $T(t)$).

Proposition 3.1 (Stationary cache occupancy).

$$\pi := \lambda E\left[\int_0^X (1 - T(t))dt\right]. \tag{3}$$

Proof. *Let V be the time during which the document is in the cache between two consecutive request arrivals. We have $\pi = E[V]/E[X] = \lambda E[V]$ by renewal theory. Let us find $E[V]$. Define the binary rv $U(t)$ to be one if the document is in the cache at time t and zero otherwise. Without loss of generality consider the interval $[0, X]$ corresponding to the inter-arrival time between the first and the second request. We have*

$$\begin{aligned} E[V] &= E\left[\int_0^X U(t)dt\right] = E_X\left[\int_0^X E[U(t)|X]dt\right] = \\ &= E_X\left[\int_0^X (1 - T(t))dt\right] \end{aligned}$$

where the last equality follows from $E[U(t)|X] = E[U(t)] = P(U(t) = 1) = P(T > t)$. ◇

Proposition 3.2. . *The CDF $G(t)$ of inter-miss times is the unique bounded solution of the integral equation*

$$G(t) = \int_0^t G(t-x)\bar{T}(x)dF(x) + \int_0^t T(x)dF(x). \tag{4}$$

Proof. Let X_1 (resp. Y_1, T_1) denote the first inter-arrival time (resp. first inter-miss time, first TTL) after $t = 0$. Since $Y_1 \geq X_1$, the event $\{Y_1 < t\}$ may only occur if $X_1 < t$. Therefore,

$$\begin{aligned} G(t) &= P(Y_1 < t, X_1 < t, X_1 \leq T_1) \\ &\quad + P(Y_1 < t, T_1 < X_1 < t) \\ &= P(Y_1 < t, X_1 < t, X_1 \leq T_1) + P(T_1 < X_1 < t) \quad (5) \\ &= P(Y_1 < t, X_1 < t, X_1 \leq T_1) + \int_0^t T(x)dF(x) \quad (6) \end{aligned}$$

where (5) follows from the fact that the event $\{Y_1 < t\}$ is true when $T_1 < X_1 < t$. It remains to evaluate the probability $P(Y_1 < t, X_1 < t, X_1 \leq T_1)$ in (6). By conditioning on X_1 and T_1 we obtain

$$\begin{aligned} &P(Y_1 < t, X_1 < t, X_1 \leq T_1) = \quad (7) \\ &\quad \int_{x=0}^t \int_{\tau=x}^\infty G(t-x)dF(x)dT(\tau) \\ &= \int_0^t G(t-x)(1 - T(x))dF(x), \quad (8) \end{aligned}$$

where the first equality is due to the fact that the TTL is renewed at each request and then $Y_1 - X_1$ conditioned to $X_1 \leq T_1$ has the same distribution of Y_1.

Suppose that there are two solutions $G_1(t)$ and $G_2(t)$ satisfying (4). Then $G_1(t) - G_2(t) = \int_0^t (G_1(t) - G_2(t))(1 - T(x))dF(x)$. By Laplace transforming both sides of this equality, it appears evident that $G_1^*(s) - G_2^*(s) = 0$ and then the solution is unique. ◇

If TTLs are exponentially distributed with rate μ, then

$$\pi = \frac{\lambda(1 - F^*(\mu))}{\mu} \tag{9}$$

from (3). Given that $T(t) = 1 - e^{-\mu t}$, $h_P = F^*(\mu)$ from (1), which in turn implies from (2) that $h_R = \lambda F^*(\mu)$ and $m_R = \lambda(1 - F^*(\mu))$. Taking the Laplace transform of both sides of (4) yields

$$G^*(s) = \frac{F^*(s) - F^*(s+\mu)}{1 - F^*(s+\mu)}. \tag{10}$$

We can check from (10) that $m_R = -(dG^*(s)/ds|_{s=0})^{-1}$.

If the TTL is a constant, equal to T, (4) becomes

$$G(t) = \int_0^{t\wedge T} G(t-x)dF(x) + (F(t) - F(T))\mathbf{1}(t > T) \tag{11}$$

with $a \wedge b = \min(a, b)$. In this case the hit probability is $F(T)$.

3.2 Relation with other replacement policies

Consider a single cache with capacity B and M contents, whose request processes are independent Poisson processes with different rates λ_k for $k = 1, 2, \ldots, M$. We shall tune the TTL policy (i.e. the timer rate μ_k for the k-th content) in order to get the same performance metrics of common replacement policies like LRU, FIFO or RND. Without loss of generality, we consider a LRU cache and we denote by p_k the corresponding stationary cache occupancy for content k. This distribution and the corresponding one for FIFO have been calculated in [18]. For the exponentially distributed TTL cache, the stationary occupancy of the k-th content is given by $\pi_k = \lambda_k(1 - F_k^*(\mu_k))/\mu_k$, where $F_k^*(s) = \lambda_k/(\lambda_k + s)$. It appears evident that, for the infinite capacity TTL cache we can always select $\mu_k = \lambda_k(1/p_k - 1)$ such that $\pi_k = p_k$. However, under storage constraints like finite capacity B, we can still take $\mu_k = \lambda_k(1/p_k - 1)$ such that $\sum_k p_k = B = \sum_k \pi_k$ as we shall see in Sec.7 for the practical TTL policy. From the equality of the stationary cache occupancies, the equality of hit/miss probabilities and rates follows thanks to the PASTA property of request processes. In this sense, the TTL policy is more general than LRU and FIFO, since it can mimic their behavior[3].

3.3 Line of caches

Consider the linear cache network in Fig.1a composed of N TTL caches labeled $1, \ldots, N$, without exogenous requests at caches $2, \ldots, N$. Requests arrive to cache 1 according to a renewal process with generic inter-arrival time X and arrival rate λ. TTLs at all caches are mutually independent and *exponentially* distributed random variables with rate μ_n at cache n.

The arrival process at cache n is the miss process at cache $n-1$ since there are no exogenous arrivals. Moreover, it is easily seen that the miss times at cache $n-1$ form regeneration points, so that the miss process at this cache, and therefore the arrival process at cache n, is a renewal process. We can then apply recursively the results obtained for a single cache. In particular, if we denote by $G_n^*(s)$ the LST of the inter-miss times at cache n, we may apply formula (10) where the

[3]We observe that the inter-miss time distributions are very different if timers are exponential, but if we select different TTL distributions we could match also them.

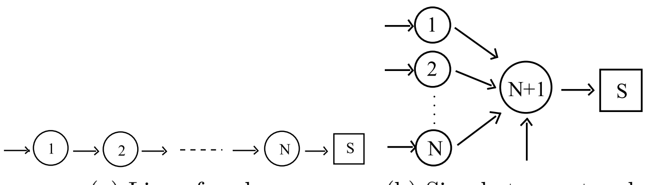

Figure 1: Network architectures

LST of the inter-arrival times is $G_{n-1}^*(s)$. We obtain

$$G_n^*(s) = \frac{G_{n-1}^*(s) - G_{n-1}^*(s+\mu_n)}{1 - G_{n-1}^*(s+\mu_n)} \tag{12}$$

for $n = 1, \ldots, N$, where $G_0^*(s) = F^*(s)$. At the cache n the hit probability $h_{P,n} = G_{n-1}^*(\mu_n)$, the miss rate $m_{R,n} = m_{R,n-1}(1 - h_{P,n})$ and hit rate $h_{R,n} = m_{R,n-1}h_{P,n}$ are derived by using (12)

$$m_{R,n} = \lambda \prod_{i=0}^{n-1} (1 - G_i^*(\mu_{i+1})), \tag{13}$$

$$h_{R,n} = \lambda \prod_{i=0}^{n-2} (1 - G_i^*(\mu_{i+1}))\ G_{n-1}^*(\mu_n), \tag{14}$$

with $m_{R,0} := \lambda$. The occupancy of cache n is given by applying (3) with $F(t) = G_{n-1}(t)$ and $\lambda = m_{R,n-1}$, hence

$$\pi_n = m_{R,n-1}\left(\frac{1 - G_{n-1}^*(\mu_n)}{\mu_n}\right). \tag{15}$$

3.4 Simple tree

Consider the tree network in Fig.1b with one root (labeled $N+1$) and N children (leaves) labeled $n = 1, \ldots, N$. The exogenous requests arrive at the n-th node according to a Poisson process with rate λ_n. While the TTL is exponentially distributed with rate μ_n at the n-th leave, we assume an arbitrary CDF $T_{N+1}(t)$, with LST $T_{N+1}^*(s)$ for the TTL at the root. Using the results in Section 3.3 to study caches $n = 1, \ldots N$ in isolation, and Theorem 2.1 to calculate the CDF of the overall request inter-arrival time at cache $N+1$, we can derive all metrics of interest [6].

4. APPROXIMATE RESULTS

The exact results in Section 3 cannot be easily extended to general networks. In fact, in the presence of exogenous requests, the aggregated (overall) arrival process at a cache is not a renewal process. Hence, we cannot apply Proposition 3.2 that allows us to characterize the miss requests process. However, we can still determine all performance metrics (e.g: see Sec. 3.4) at the cache by calculating the CDF of the inter-arrival times (Theorem 2.1). In this section we develop an approximation method to overtake this limitation. The solution derived produces highly accurate approximations under more general networks for all metrics considered in Section 3 (hit/miss probabilities, hit/miss rate, cache occupancy). The quality of the approximation is assessed in [6] and it is based on the following statement:

Approximation A1: the overall arrival process at each node is a renewal process.

As a consequence of **A1** and Prop. 3.2, we approximate the miss process at a node by a renewal process. With a slight abuse of notation we will use the notation of Section 3 to denote the corresponding approximate values calculated under Approximation **A1**. For example, $H_n(t)$ is used to denote the approximate CDF of the overall inter-arrival times and similarly $G_n(t)$, Λ_n, $m_{R,n}$, $h_{P,n}$ and $h_{R,n}$ are used to denote approximate quantities at node n. Regarding the total rate Λ_n, note that

$$\Lambda_n = \lambda_n + \sum_{i \in \mathcal{C}(n)} m_{R,i} \tag{16}$$

where $\mathcal{C}(n)$ is the set of children of node n. As in Section 3, exogenous arrivals at each node n form a renewal process, with CDF $F_n(t)$. Thanks to **A1**, we invoke Theorem 2.1 to get

$$\begin{aligned} H_n(t) &= 1 - \frac{\lambda_n}{\Lambda_n}\bar{F}_n(t) \prod_{i \in \mathcal{C}(n)} \nu_i \int_t^\infty \bar{G}_i(u)du \\ &\quad - \sum_{i \in \mathcal{C}(n)} \frac{\nu_i}{\Lambda_n}\bar{G}_i(t)\lambda_n \int_t^\infty \bar{F}_n(u)du \\ &\quad \times \prod_{\substack{j \in \mathcal{C}(n) \\ j \neq i}} \nu_j \int_t^\infty \bar{G}_j(u)du. \end{aligned} \tag{17}$$

An approximation of the CDF of the inter-miss times at cache n is obtained from Proposition 3.2

$$\begin{aligned} G_n(t) = &\int_0^t G_n(t-x)\bar{T}_n(x)dH_n(x) \\ &+ \int_0^t T_n(x)dH_n(x) \end{aligned} \tag{18}$$

where $T_n(t)$ the CDF of the TTL duration at cache n. Eqs (17)-(18) provide a recursive procedure for calculating, at least numerically, approximations of the CDFs $G_n(t)$ and $H_n(t)$ for each cache n of a general network topology, from which we can derive approximate formulas for all metrics.

However, even for small networks the numerical complexity of this procedure can be very high as it requires calculating integrals over infinite ranges (see Eq. (17)) and solving integral equations (see Eq. (18)).

In order to show that this complexity can be significantly reduced, we focus on a particular class of tree networks, class $\mathcal{N}$ and we give explicit results. The TTL at each node n is exponentially distributed with rate μ_n. A network belongs to class $\mathcal{N}$ if, in addition to **A1**, the following approximation holds:

Approximation A2: The node n is fed by the superposition of two independent request arrival processes: one (*stream 1*) is the miss rate of a child of cache n and is a generic renewal process and the other one (*stream 2*) is a renewal process with CDF of the form

$$K_n(t) = 1 - \sum_{m=0}^{M_n} \alpha_{n,m} e^{-\beta_{n,m} t} \tag{19}$$

where $0 \leq M_n < \infty$ and $\{\beta_{n,m}\}_m$ is a set of non negative numbers.

In what follows we assume without loss of generality that stream 1 originates from a cache child labeled $n-1$ and then we denote the CDF of the inter-miss times in stream 1 as $G_{n-1}(t)$ and the miss rate as ν_{n-1}. From (19) the arrival rate of stream 2 is $\eta_n := \sum_{m=0}^{M_n} \alpha_{n,m}\beta_{n,m}$, the total arrival rate at node n is $\Lambda_n = \nu_{n-1} + \eta_n$. Approximations **A1** and **A2** together yield the following procedure for approximating $G_n^*(s)$ and $H_n^*(s)$.

PROPOSITION 4.1 (APPROXIMATION FOR CLASS $\mathcal{N}$). *Under Approximations* **A1** *and* **A2***, for each node n,*

$$\begin{aligned} H_n^*(s) = 1 - s\frac{\eta_n}{\Lambda_n} &\sum_{m=0}^{M_n} \frac{\alpha_{n,m}}{s+\beta_{n,m}} \\ -s^2\frac{\eta_n \nu_{n-1}}{\Lambda_n} &\sum_{m=0}^{M_n} \frac{\alpha_{n,m}(1-G_{n-1}^*(s+\beta_{n,m}))}{(s+\beta_{n,m})^2\beta_{n,m}} \end{aligned} \tag{20}$$

and

$$G_n^*(s) = \frac{H_n^*(s) - H_n^*(s+\mu_n)}{1 - H_n^*(s+\mu_n)}. \tag{21}$$

Proof.

$$\begin{aligned} H_n(t) = 1 - &\frac{\eta_n \nu_{n-1}}{\Lambda_n} \sum_{m=0}^{M_n} \alpha_{n,m} \\ &\times \left(\bar{G}_{n-1}(t) \int_t^\infty e^{-\beta_{n,m}u}du + e^{-\beta_{n,m}t} \int_t^\infty \bar{G}_{n-1}(u)du \right) \end{aligned}$$

from which we deduce (20). (21) is obtained from (10). $\diamond$ The hit probability is simply $h_{P,n} = H_n^*(\mu_n)$, the total arrival rate $\Lambda_n = \nu_{n-1} + \eta_n$ and the miss rate $\nu_n = \Lambda_n(1 - H_n^*(\mu_n))$ are

$$\nu_n = \sum_{i=1}^{n} \eta_i \prod_{j=i}^{n} (1 - H_j^*(\mu_j)), \tag{22}$$

$$\Lambda_n = \sum_{i=1}^{n-1} \eta_i \prod_{j=i}^{n-1} (1 - H_j^*(\mu_j)) + \eta_n, \tag{23}$$

Relations (20)-(21) and (22)-(23) provide a recursive procedure for calculating Λ_n and $H_n^*(\mu_n)$ for each n, from which we obtain approximations for the hit probability, hit rate, miss rate and stationary occupancy at node n:

$$\begin{aligned} h_{P,n} &= H_n^*(\mu_n),\ h_{R,n} = \Lambda_n H_n^*(\mu_n), \\ m_{R,n} &= \Lambda_n(1 - H_n^*(\mu_n)),\ \pi_n = \Lambda_n\left(\frac{1 - H_n^*(\mu_n)}{\mu_n}\right). \end{aligned} \tag{24}$$

The latter result follows from (9).

In [6], we show that the network in Fig. 2 belongs to the class $\mathcal{N}$, when TTLs are exponentially distributed and exogenous request processes are Poisson.

5. VALIDATION

In this section we investigate the accuracy of the approximation method developed in Section 4. Recall that the method consists in assuming that all internal arrival processes at a node (i.e. processes formed of the miss processes of the

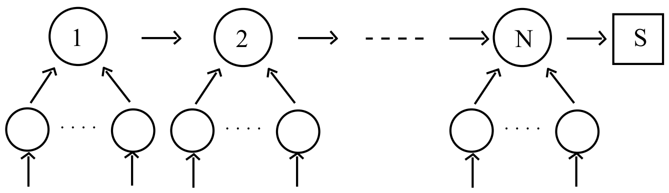
Figure 2: Line of simple tree networks

node's children) are renewal processes and to use Eq. (17) to calculate the CDF of the inter-arrival times of the superposed process. The miss process at this node can then be characterized by using Eq. (18) and the procedure is repeated at the node's parent.

We focus our validation on the case when the TTLs are exponentially distributed, but we also provide some results for constant TTLs.

5.1 Exponential timers

We start by observing that it is possible to model a class $\mathcal{N}$ network with N caches as an irreducible Markov process, with state $\mathbf{x}(t) = (x_1(t), \dots, x_N(t)) \in \mathcal{E} = \{0,1\}^N$, where $x_n(t) = 1$ (resp. $x_n(t) = 0$) if the document is present (resp. missing) at time t at node n. Once the steady-state probabilities $(p(\mathbf{x}))$ have been calculated, the exact values of the performance metrics of interest can be obtained by conveniently combining the stationary probabilities and the rates. For example the stationary occupancy of cache i is $\pi_i^M = \sum_{\mathbf{x}\in\mathcal{E}, x_i=1} p(\mathbf{x})$ (the superscript "M" stands for "Markov") . For a line of caches the hit probability and the miss rate at cache 1 are respectively $h^M_{P,1}(1) = p(1,*)$ and $m^M_{R,1} = \lambda_1 p(0,*)$, while for cache 2 it holds

$$h^M_{P,2} = \frac{\lambda_1 p(0,1,*) + \lambda_2(p(0,1,*) + p(1,1,*))}{\lambda_1(p(0,0,*) + p(0,1,*)) + \lambda_2}$$

and $m^M_{R,2} = \lambda_1 p(0,0,*) + \lambda_2(p(0,0,*) + p(1,0,*))$, where $p(i,*) = \sum_{x_2,\dots,x_N\in\{0,1\}} p(i,x_2,\dots,x_N)$ and $p(i,j,*) := \sum_{x_3,\dots,x_N\in\{0,1\}} p(i,j,x_3,\dots,x_N)$ are the stationary probabilities that cache 1 is in state $i \in \{0,1\}$ and caches $(1,2)$ are in state $(i,j) \in \{0,1\}^2$, respectively. Due to space constraints we omit the general expressions for these quantities for a generic cache in the line and those for a line of simple tree networks that can be similarly calculated.

In the rest of this section we compare our approximate results versus the exact ones that can be obtained studying the Markov process. A comparison of the computational costs of the two approaches is in Section 6. We consider first the line network in Fig. 3: it has four nodes ($N = 4$), exogenous arrivals and exponentially distributed TTLs at node n with rate λ_n and μ_n, respectively. We have calculated

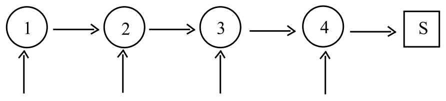
Figure 3: Line of four caches

the absolute relative errors at cache n for the hit probability $(E_{HP,n})$, the miss rate $(E_{MR,n})$ and the occupancy probability $(E_{OP,n})$. The exact value is calculated through the analysis of the Markov process, e.g. $E_{HP,n} := |h^M_{P,n} - h_{P,n}|/h^M_{P,n}$. Fig. 4 shows the CDFs of the relative errors at cache 4 for 1001 different parameter vectors $((\lambda_n, \mu_n), n = 1, \dots, 4)$. The values of the exogeneous arrival rates (resp. TTL rates) have been selected in the interval $[0.001, 10]$ (resp. $[0.1, 2]$) according to the FAST (Fourier Amplitude Sensitivity Test) method (see [21, Sec. VI-C] and references therein). We can observe that the approximation is very accurate: in 99% of the different parameter settings the relative error is smaller than 2×10^{-5}.

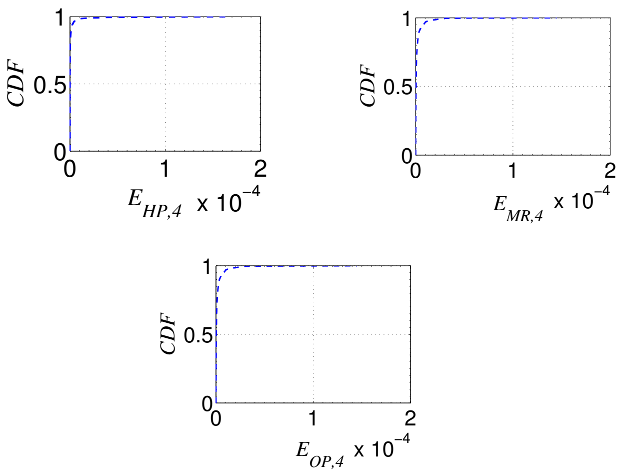

Figure 4: CDF of $E_{HP,4}, E_{MR,4}, E_{OP,4}$ for network in Fig.3

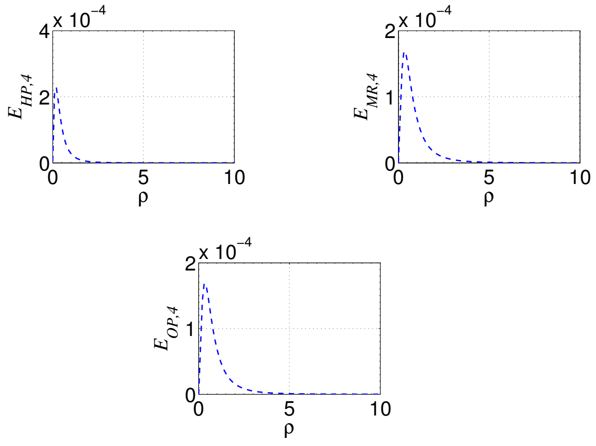

Figure 5: $E_{HP,4}, E_{MR,4}, E_{OP,4}$ for network in Fig.3 with homogeneous nodes $(\lambda_n = \lambda = \rho\mu = \rho\mu_n)$

Fig. 5 shows how the error changes for different request loads. In this case we have considered the homogeneous scenario where all the caches have the same TTL and the same exogenous arrival rate, i.e. $\mu_n = \mu$ and $\lambda_n = \lambda$ for each n. The error is shown as a function of the normalized load $\rho = \lambda/\mu$. We can observe that the largest error (about 2×10^{-4}) is obtained when arrival rates and timer rates have comparable values ($\rho \approx 1$). In this case the different request processes superposed at a node have similar time scales and

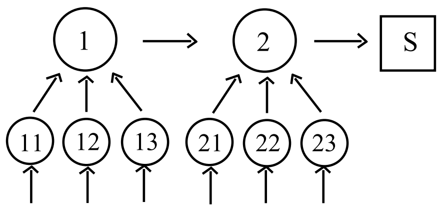

Figure 6: Linear tree network

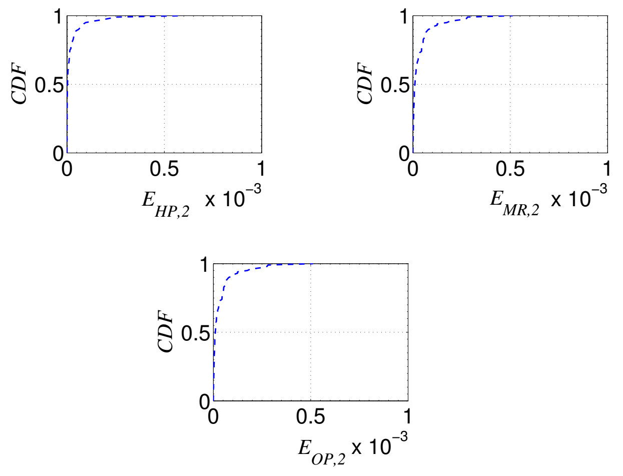

Figure 7: CDF of $E_{HP,2}$, $E_{MR,2}$, $E_{OP,2}$ for network in Fig. 6

then the inter-arrival times of the overall request process are more correlated (see also comments below).

We have also investigated the accuracy of the approximation for the line of simple tree networks (defined in Section 3.4) shown in Fig. 6: where nodes $11, 12, 13$ (resp. nodes $21, 22, 23$) have identical arrival rates and identical TTLs rates. Since the approximation results are exact for all nodes but node 2 we only report results for that node. The empirical CDFs of $E_{HP,2}$, $E_{MR,2}$ and $E_{OP,2}$ are shown in Fig. 7. Like for Fig. 4 exact results have been obtained by considering the Markov process associated with this line of simple trees network. Different request and TTL rates have been selected according to the FAST method respectively in the intervals $[0.001, 10]$ and $[0.1, 2]$. We used 4921 samples for each rate. Results are analogous to those for a line of caches. The relative errors can be larger in this scenario, but they are probably negligible for most of the applications ($< 3 \times 10^{-4}$ in 99% of the cases). We have also considered the homogenous scenario also for this topology, the relative errors have the same order of magnitude ($< 10^{-3}$).

We have shown that Assumption **A1** leads to very accurate results when exogenous arrival processes are Poisson and TTL are exponentially distributed. This let us think that the superposition of the request arrival processes at every cache is very 'close' to a renewal process. In order to justify such statement, we have calculated the first autocorrelation lag (r_1) for the actual arrival process at node 2 in Fig. 3 using Eq. (6.4) in [22]. This autocorrelation lag depends on the

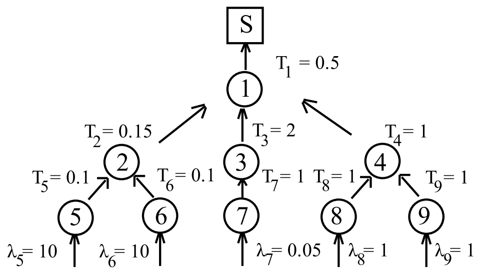

Figure 8: Tree network

arrival rates λ_1 and λ_2 and the timer μ_1. We have found that for any possible choice of these parameters $0 > r_1 > -0.015$. Simulation results show that the autocorrelation is even less significant at larger lags. We can then conclude that the inter-arrival times are weakly coupled.

5.2 Deterministic Timers

When timers are deterministic we need to rely on the general procedure described in Section 4 and based on Eqs (17) and (18). There are two sources of errors in this procedure. Firstly, the aggregate request process at a cache is not a renewal process and it is not correct to apply the renewal equation (18). Secondly, both the steps (17) and (18) introduce some numerical errors. Two parameters determine the entity of the numerical error: 1) the time interval (τ) from which the CDF samples are taken, 2) the time distance between two consecutive samples (Δ). Clearly the larger τ and the smaller Δ the smaller the numerical error, but also the larger the computational cost.

We have implemented a Matlab numerical solver that iteratively determines the CDFs in the network as described above, and then the metrics of interest for a cache network. The integrals appearing in Eqs. (17) and (18) are approximated as simple sums and for simplicity the same values τ and Δ have been considered for all the CDFs. These parameters are selected as follows: our solver first obtains an approximated solution for the whole network assuming that all the request processes are Poisson and set the parameter τ to 5 times the largest expected inter-arrival time in the network. The parameter Δ is set to one thousandth of the minimum of the TTL values and the expected interarrival times of the exogenous request processes.

We present some preliminary results for the tree network of Fig. 8 in the following situation: infinite capacity caches, asymmetric traffic conditions and non-exponential (i.e constant) life times. More specifically, the exogenous request processes are Poisson processes with rates λ_i ($i = 5, 6, 7, 8, 9$) and TTL values are T_i ($i = 1, 2, \ldots 9$). In order to evaluate the relative error of the estimated metrics, we have considered as correct values those obtained through a long simulation. For example if our method predicts the value $h_{P,n}$ for the hit probability rate at node n and the 99% confidence interval, calculated by simulation, is $[h^S_{P,n} - \epsilon, h^S_{P,n} + \epsilon]$, the relative error is calculated as $|h_{P,n} - h^S_{P,n}|/h^S_{P,n}$. The relative incertitude of the simulation ($\epsilon/h^S_{P,n}$) is at most 0.3×10^{-4}. For all the performance metrics and all the caches the relative error of our approach is less than 10^{-2}.

6. COMPUTATIONAL COST

In this section we perform a preliminary analysis of the computational cost of our approach.

We first address the case of a class $\mathcal{N}$ network, and in particular we consider a line of simple tree networks with N trees and M nodes in total as in Fig. 2. Since the computational cost for all the metrics is roughly the same, we focus here on the hit probability. In order to calculate the hit probability at one of the roots of the simple trees, say it cache n, we need to evaluate the LST $H_n^*(\mu_n)$ (Eq. (24)). This requires a number of operations proportional to the number of children of cache n (R_n) and the evaluation of the LST of the miss rate coming from cache $n-1$ in μ_n, i.e $G_{n-1}^*(\mu_n)$ (Eq. (20)). In turn $G_{n-1}^*(\mu_n)$ can be calculated evaluating $H_{n-1}^*(s)$ in two points (μ_n and $\mu_n + \mu_{n-1}$) (Eq. (21)) and so on recursively. This implies that the cost to calculate the hit probability at cache n is $O(\alpha R_n + 2^n)$ for some constant α. When evaluating the hit probability at other caches the same LSTs needs to be evaluated, but in general at different points, then we have that the total cost is $O(\sum_{n=1}^{N} \alpha R_n + 2^n) = O(\alpha M + 2^N)$. Then, depending on the topology of the network, the cost can be mainly linear in the number of nodes (for a network with small depth, e.g. when there are a few trees each with a lot nodes) or exponential in the number of nodes (for a network with large depth, e.g. for the linear network in Fig. 1).

It is interesting to compare this cost with alternative approaches. For the line of simple tree networks, all the metrics can be exactly calculated solving a Markov process as we mentioned in Section 5. The size of the state space is 2^M, then the cost of determining the steady-state distribution by solving the linear equation system is $O(2^{3M})$ and this is much larger than the cost of our method $O(\alpha M + 2^N)$. A different approach is to obtain an approximated steady-state distribution of the Markov process using an iterative method. This approach takes advantage of the fact that most of the transition rates have value zero. In fact a state change is triggered by an exogenous request arrival at a cache that does not have the data or by a timer expiration at a cache with the data, i.e. from a given state we can only reach other M states. Then the number of non-zero rates is equal to $M2^M$ and each iteration of the method requires $O(M2^M)$ operations. The total cost of the iterative method is then $O(KM2^M)$, where K is the number of iterations until termination and depends on the spectral gap of the matrix used at each iteration and on the required precision, but in general we can expect $O(KM2^M) << O(2^{3M})$. Assuming that this is the case, we can observe that our method, even in the worst case of the linear network, is still more convenient than solving the Markov process, because $O(2^M) < O(KM2^M)$.

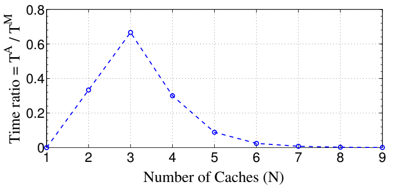

Figure 9: Running time comparison

Fig. 9 shows the ratio of the computation times to calculate our approximation (T^A) and to solve the Markov chain (T^M) for a line of N caches (with $N = 1, 2, \ldots 9$). Both the methods have been implemented in Matlab, in particular the function linsolve has been used to determine the steady-state distribution of the Markov chain.

Let us now consider the case of a general tree network with constant TTLs (equal to T). In this case there is no exact solution to compare our approach with, so we consider simulations as an alternative approach. We perform an asymptotic analysis. A meaningful comparison of the computational costs needs to take also into account the incertitude of the solution: both the simulations and our method can produce a better result if one is willing to afford a higher number of operations. In order to combine these two aspects in our analysis we consider as metric the product precision times number of operations. Intuitively the larger this product the more expensive is to get a given precision. For the simulations the computational cost is at least proportional to the number of events that are generated, let us denote it by n_E. The incertitude on the final result can be estimated by the amplitude of the confidence interval, that decreases as $1/\sqrt{n_E}$, then the product precision times number of operations is proportional to $\sqrt{n_E}$ for the simulations. In the case of our approach, the heaviest operation is the solution of the renewal equation. If we adopt the same τ and Δ for all the integrals, we need to calculate the value of the CDF of the miss rate ($G(t)$) in $n_P = \tau/\Delta$ points and then we need to calculate n_P integrals. The integration interval is at most equal to the TTL duration T (see Eq. 11), then each integral requires a number of operations proportional to $n'_P = T/\Delta$. If the value of τ is selected proportionally to T, then the cost of our method is proportional to n_P^2. A naive implementation of the integral as a sum of the function values leads to an error proportional to the amplitude of the time step and then inversely proportional to n'_P or n_P. In conclusion the product precision times the number of operations is proportional to n_P. Then, for a given precision, our method would require a number of points much larger than the number of events to be considered in the corresponding simulation (at least asymptotically). The comparison would then lead to prefer the simulations at least when small incertitude is required (then large n_E and n_P). In reality integrals can be calculated in more sophisticated ways, for example if we adopt Romberg's method, with a slightly larger computation cost, we can get a precision proportional to n_P^{-2}. In this case the product precision times number of operations is a constant for our method, that should be preferred.

7. A PRACTICAL TTL POLICY

While the model considered above allows for an arbitrarily large number of contents, a real cache will have a finite capacity B. In this section, we consider a possible practical implementation (TTL Impl) of the ideal TTL policy (TTL Model) we have studied above. The cache uses timers for each content as described above but it does not discard contents whose timer has expired as long as some space in the buffer is available. If a new content needs to be stored and the cache is full the content to be erased is the content whose timer has expired since a longer time (if any) or the content whose timer is going to expire sooner.

We have compared the performance of the TTL Impl with that of our TTL Model (in Sec. 3.3) for a line of two caches with same capacities $B_1 = B_2 = 20$ where the requests for each content $k = 1, \dots, M = 200$ arrive only at the first cache. The request processes are independent Poisson processes with rates distributed according to a Zipf law with parameter $\alpha = 1.2$. The timers are i.i.d. exponential random variables with rate μ_1 and μ_2 respectively at the two caches. Each timer rate has been selected so that the expected buffer occupancy for both the TTL Impl and the TTL Model equals the corresponding cache capacity. In other words, μ_n at cache $n = 1, 2$ is chosen such that $\sum_{k=1}^{M} \pi_{n,k} = B_n$. Results show that the performances of the two policies are very similar. The aggregate hit probabilities at the two caches are respectively $h_{P,1} = 0.5584$ and $h_{P,2} = 0.4663$ for the TTL Model and $\hat{h}_{P,1} = 0.5618$ and $\hat{h}_{P,2} = 0.4124$ for the TTL Impl. At the content granularity, the matching is even accurate at cache 1, so that we just report in Fig. 10 the results at cache 2. These preliminary results suggest that our analysis can be useful to study TTL policies when caches are constrained.

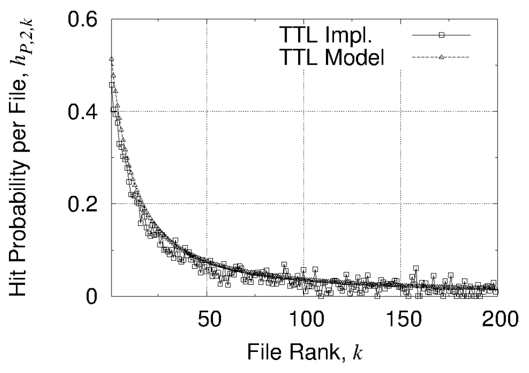

Figure 10: TTL-policy model and its implementation

8. CONCLUSION

In this paper, we introduced a novel Time-To-Live replacement policy for buffering routers of ICN and we have developed a set of building blocks for the performance evaluation of TTL cache networks based on simple renewal arguments. We derived exact results and closed form formulas to assess the metrics of some hierarchical caching networks like the linear and the simple tree networks. We also provided a recursive and approximate procedure to study a general network topology. For a large class of tree networks with exponentially distributed and deterministic TTLs, we showed that our approximation is highly accurate with relative errors of 10^{-3} and 10^{-2} respectively. Our approach scales easily and it is promising since it appears as a simple unifying model for accurately modeling a richer class of networks such as heterogeneous caching networks under different replacement policies. We have also demonstrated that our TTL model can be implemented and used to optimize a multi-content cache network under realistic constraints such as the cache size limitation. Ongoing research is investigating the effect of finite capacity cache networks, how to reduce the computational cost of our approach in the case of constant TTLs and how to take into account correlated request processes by using semi-Markov processes.

9. ACKNOWLEDGEMENTS

This research has been supported by Orange Labs in the framework of the contract "Performance models for CCN architecture". We thank in particular Orange Labs researchers Bruno Kauffmann, Luca Muscariello and Alain Simonian for the stimulating discussions and for the valuable feedback.

10. REFERENCES

[1] J. R. Bitner, "Heuristics that monotonically organize data structures", *SIAM J. Computing*, **8**, pp. 82-110, 1979.

[2] P. J. Burville and J. F. C. Kingman, "On a model for storage and search", *Journal of Applied Probability*, **10**, pp. 697-701, 1973.

[3] J. Mc Cabe, "On serial files with relocable records", *Oper. Res.*, **13**, pp. 609-618, 1965.

[4] G. Carofiglio, M. Gallo, L. Muscariello and D. Perino, "Modeling data transfer in content-centric networking", *Proc ITC23*, San Francisco, CA, USA, Sep. 6-8, 2011.

[5] H. Che, Y. Tung and Z. Wang, "Hierarchical Web caching systems: modeling, design and experimental results", *IEEE J. on Selected Areas in Communications*, Vol. 20, No. 7, pp. 1305-1314, Sep. 2002.

[6] N. Choungmo Fofack, P. Nain, G. Neglia, D. Towsley, "Analysis of TTL-based Cache Networks", *INRIA Research Report RR-7883*, 2012.

[7] E. G. Coffman Jr. and P. Jelenkovic, "Performance of the Move-to-Front algorithm with Markov-modulated request sequences", *Operations Research Letters*, **25**, pp. 109-118, 1999.

[8] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes", *Proc. ACM Sigmetrics 1990*, Boulder, CO, USA, May 22-25, 1990, pp. 143-152.

[9] R. P. Dobrow and J. A. Fill, "The move-to-front rule for self-organizing lists with Markov dependent requests", *Discrete Probability and Algorithms*, IMA Volumes in Mathematics and its Applications, D. Aldous, P. Diaconis, J. Spencer, and J. M. Steele (Eds), **72**, pp. 57-80, Springer-Verlag, 1995.

[10] P. Flajolet, D. Gardy and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search", *Discrete Applied Mathematics*, **39**, pp. 207-229, 1992. First version appeared as an INRIA Tech. Report, No. 720, Aug. 1987.

[11] W. J. Hendricks, "The stationary distribution of an interesting Markov chain", *Journal of Applied Probability*, **9**, pp. 231-233, 1972.

[12] J. A. Fill, "Limits and rate of convergence for the distribution of search cost under the move-to-front rule", *Theoretical Computer Science*, **176**, pp. 185-206, 1996.

[13] P. Jelenkovic, "Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities", *The Annals of Probability*, Vol. 9, No. 2, pp. 430-464, 1999.

[14] P. Jelenkovic and A. Radovanović, "Least-recently used caching with dependent requests", *Theoretical Computer Science*, **326**, pp. 293-327, 2004.

[15] P. Jelenkovic and A. Radovanović and M. Squillante, "Critical sizing of LRU caches with dependent requests", *Journal of Applied Probability*, Vol. 43, No. 4,

pp. 1013-1027, December 2006.

[16] J. Jung, A. W. Berger and H. Balakrishnan, “Modeling TTL-based Internet Caches”, *Proc. IEEE Infocom 2003*, San Francisco, CA, USA, Mar. 30 - Apr. 3, 2003.

[17] J. Jung, E. Sit, H. Balakrishnan and R. Morris, “DNS performance and the effectiveness of caching”, *Proc. ACM SIGCOMM Workshop on Internet Measurement (IMW ’01)*, New York, NY, USA, Nov. 1-2, 2001.

[18] W. F. King, “Analysis of demand paging algorithm”, *Information Processing*, Vol. 71, pp. 485-490, 1972. [Appeared first under the same title as IBM Research Report, RC 3288, Mar. 17, 1971.]

[19] T. Lai and H. Robbins, “Asymptotically efficient adaptive allocation rules”, *Advances in Applied Mathematics*, Vol. 6, pp.4-22, 1985.

[20] N. Laoutaris, S. Syntila and I. Stavrakakis, “Meta Algorithms for Hierarchical Web Caches”, *IEEE IPCCC 2004*, Phoenix, Arizona, April 2004.

[21] I. Lassoued, A. Krifa, C. Barakat and K. Avrachenkov, “Network-wide monitoring through self-configuring adapative system”, *Proc. IEEE Infocom 2011*, Shanghai, China, Apr. 10-15, 2011.

[22] A. T. Lawrance, “Dependency of intervals between events in superposition processes”, *Journal of the Royal Statistical Society*, Series B (Methodological), Vol. 35, No. 2, pp. 306-315, 1973.

[23] E. J. Rosensweig, J. Kurose and D. Towsley, “Approximate models for general cache networks”, *Proc. IEEE Infocom 2010*, San Diego, CA, USA, Mar. 15-19, 2010.

[24] V. Jacobson, D. K. Smetters, J. D. Thorntorn, M. Plass, N. Briggs and R. L. Braynard, “Networking named content”, *Proc. ACM CoNEXT 2009*, Rome, Italy, Dec. 1-4, 2009.

[25] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble and M. Levy, “An analysis of Internet content delivery systems”. *SIGOPS Operating System Review*, Vol. 36, isse SI, pp. 315-327, 2002.