On the Performance of General Cache Networks

N. Choungmo Fofack
Orange Labs, France
nicaise.choungmofofack@orange.com

M. Dehghan
Univ. Massachusetts Amherst
mdehghan@cs.umass.edu

D. Towsley
Univ. Massachusetts Amherst
towsley@cs.umass.edu

M. Badov
Univ. Massachusetts Amherst
mbadov@cs.umass.edu

D. L. Goeckel
Univ. Massachusetts Amherst
goeckel@ecs.umass.edu

ABSTRACT

The performance evaluation of cache networks has gain a huge attention due to content-oriented delivery technologies. If general network topologies are more realistic than hierarchical networks widely studied in the literature, their analysis is significantly challenging. Existing models mainly focus on trees where content custodians are located at the root and the one-way child-to-parent request forwarding schema is common. In this paper, we consider complex and irregular networks where requests may flow possibly in opposite directions from/to several sources/destinations. Moreover, we assume that caches may run one of Time-To-Live (TTL)-based policies recently introduced for content-centric networks and modern Domain Name System [5]. We then derive an analytical framework and a polynomial-time algorithm that approximate accurately performance metrics of arbitrary graph-based and heterogeneous TTL-based cache networks. Simulations show that our simplified methodology may accurately predict metrics of interest on networks of caches running popular replacement algorithms (e.g. LRU, FIFO, or Random) without restricting its scope of application to this interesting use case. Unlike existing approaches, ours scales as network and content catalog sizes increase.

Categories and Subject Descriptors

H.4 [**Content-oriented Networking**]: Performance

Keywords

Performance Analysis, Approximation Algorithms, Cache Networks, LRU, FIFO, Random, Time-To-Live (TTL)

1. INTRODUCTION

Over the past years, isolated data repositories running popular cache replacement algorithms have received significant attention. Nowadays, interest has shifted from caching systems in isolation to interconnected caches. The benefits of the latter approach come from storing contents in caches that are universally deployed or distributed across the network. This trend is mostly driven by the recent development of content-oriented technologies [1, 12]. The purpose is to adapt the network architecture to accommodate the current content usage patterns (Video-on-Demand, User-generated contents, etc.) with the potential to reduce congestion, improve content delivery speed as networks increase in size.

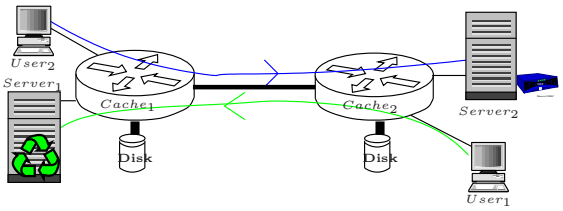


Figure 1: Tandem of two caches with bidirectional flows

Network performance goals can be achieved in different ways. We focus on content distribution technologies [1] which allow *on-demand* caching through deployment of caches at various locations. Briefly speaking, when a data item is first requested, it is temporarily stored at some nodes and subsequent requests are served directly from these local copies. Therefore, users may experience a better quality of service. Obviously, the benefits of such in-network caching choices are highly tied to the performance of the underlying cache networks. These performance metrics can be significantly challenging to predict since the multi-cache systems that arise from content-oriented architectures are more complex, irregular, and heterogeneous. More precisely, these networks may have arbitrary topologies and their nodes may show different features (policies, capacities, routing tables, etc.).

When requests of all files share the same routing topology with content custodians of all files located at the same root nodes, bidirectional flows as described in Fig. 1 cannot occur. In this very specific case, requests flow in the same direction from child nodes to parents towards the root(s) and the network topology reduces to all nodes belonging to this routing topology or directed acyclic graph. We refer to this case as the *unique routing topology model*. Instances of this routing model have been studied in the literature on networks with linear [14], tree [4, 13], polytree [6, 15], feed-forward topologies [3, 5, 8]. However, their approaches rely heavily on the unique routing topology model which is here a tree/hierarchical structure and the one-way child-to-parent forwarding schema of the network. This strong requirement

VALUETOOLS 2014, December 09-11, Bratislava, Slovakia

DOI 10.4108/icst.valuetools.2014.258168

makes their models not applicable to more general cases. Unlike, Algorithm 1 presented in Section 4 do not suffer from this limitations.

As reported in [12], only a-NET model, introduced in [16], addressed so far the performance of the tandem of two caches shown in Fig. 1 where requests are routed in opposite directions. This basic network is the atom of arbitrary interconnected caches. Unfortunately, [16, 12] report relative errors of 15% and find that the violation of the Independent Reference Model (a.k.a. IRM or Poisson assumption) on miss streams of caches and aggregated request processes in the network is the major source of inaccuracy.

In this paper, we develop an analytical framework to assess performance metrics of cache networks by leveraging the concept of *Time-To-Live (TTL)-based models* recently introduced in [5] on the one hand. On the other hand, we apply two-moment matching techniques to fit hyper/shifted exponential renewal processes to general request processes [17]. Our main results are
• an analytical and extensible framework that provides computationally efficient two-moment approximations of filtered, split, and merged request streams in the network;
• a polynomial-time algorithm to approximate metrics of interest on large and heterogeneous graph-based networks;
• *Little's Law* for caching systems and bounds of the TTL-based cache characteristic time are derived;
• event-driven simulation results showing that our framework is more accurate than the existing model [16].

This paper is organized as follows. In Section 2, we describe the system under analysis and state our problem. In Section 3, we derive a *Little's Law*-like and *characteristic equations* enabling caching systems to be described via TTL-based models. We then present our algorithm to calculate characteristic times of isolated caches and our two-moment matching techniques to characterize filtered, split, and aggregated request streams. We describe our polynomial-time algorithm for network analysis in Section 4, show its accuracy in Section 5, and summarize our findings in Section 6.

2. MODEL

In this section, we state our problem on our toy network (Fig. 1). Then, we introduce the notation and assumptions that are used in our cache network model.

2.1 Problem formulation

Two sets of files are requested on the tandem of two caches of Fig. 1. User u_n requests the K_n files stored permanently at server s_n. User u_1's missed requests at cache 2 are routed towards server s_1 through cache 1. The overall request arrival process at cache 1 is formed by (well-known) exogenous requests coming from user u_2 and unknown miss processes of cache 2 and vice versa. Consequently, states of both caches become dependent. This would not occur if both servers were located at cache 2 as considered in [3, 4, 6, 13, 14, 15]. Therefore, these existing models are limited since overall request processes at both caches are unknown and the strong assumptions (i.e. tree-based structure with content custodians at located at the root and the one-way child-to-parent request forwarding schema in the network) required by their cache-by-cache iterative models do not hold.

2.2 Network model

Let $\mathcal{G} = (V, E)$ be the graph representing a general and heterogeneous cache network, $V = \{v_1, \ldots, v_N\}$ the set of caches, and $E \subseteq V \times V$ the set of connections between caches. Additionally, the file catalog is denoted by $\mathcal{F} = \{f_1, \ldots, f_K\}$. Each file is stored permanently at one or more public servers attached to nodes in the network and there is at least one path between each pair of cache–server.
Requests arrive exogenously and directly from users to some of the nodes $\{v_n \in V\}$ in this system. When a request for file f_i arrives at a cache, it generates a cache hit if the file is located in the cache and a miss otherwise. In the latter case, the request is forwarded to other caches in the network based on the routing table at each cache, until the file is located in a cache or at the server storing the file. Then the file is forwarded along the reverse path taken by the request, and stored at each cache along the way. If the cache is full when a miss occurs, one of the files in the cache is selected based on an eviction policy to make room for the new file.

2.3 Workload model

We denote by $\mathcal{R}_{n,i} = \{t_k(n,i)\}_{k\geq 0}$ the overall request process of file f_i at cache v_n where $t_k(n,i)$ is the arrival instant of the k+1-th request. $\mathcal{R}_{n,i}$ is formed by the superposition of an exogenous request process $\mathcal{E}_{n,i}$ (generated by local users of cache n), if any, and the endogenous request processes (generated by misses of other caches connected to cache n). Let $\lambda_{n,i}$ be the rate of exogenous arrivals at cache v_n, if any, and $\Lambda_{n,i}$ the intensity of $\mathcal{R}_{n,i}$. In this paper, we assume that

ASSUMPTION 1 (RENEWAL). *Exogenous request processes $\{\mathcal{E}_{n,i}, \forall n, i\}$ are renewal processes. Moreover, $\{\mathcal{E}_{n,i}, \forall i\}$ are independent at cache n.*

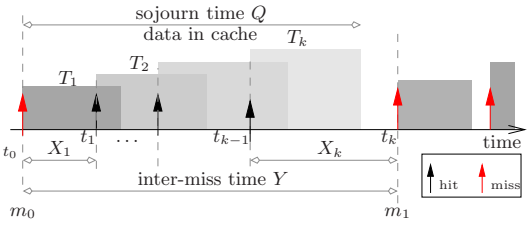
At cache v_n and for file f_i, we denote by $X_{n,i}$ the generic inter-arrival time of request in the process $\mathcal{R}_{n,i}$, $F_{n,i}(t) = P(X_{n,i} < t)$ its Cumulative Distribution Function (CDF), $\hat{F}_{n,i}(t)$ the CDF of its survival time, $N_{n,i}(t)$ the counting process, $R_{n,i}(t) = E[N_{n,i}(t)]$ the Renewal Function associated to $F_{n,i}(t)$, and $F^*_{n,i}(s) = E[e^{-sX_{n,i}}]$ its Laplace-Stieltjes Transform (LST) for all $t \geq 0$ and $s \geq 0$.

2.4 Cache model

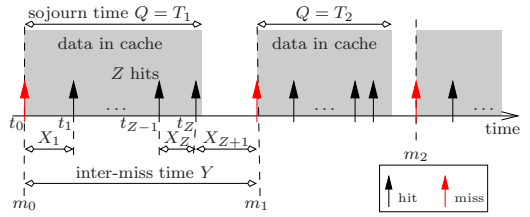
In this work, nodes of our network are endowed with a cache running a replacement algorithm that can be described with *Renewing* or *Non-renewing* TTL-based models introduced in [6] and [15] respectively. The renewing TTL-based model assigns a random value to the timer T_1 of a file at cache miss instant t_0 and later redraws this timer T_k at each cache hit instant t_{k-1} as shown in Fig. 2(a). The non-renewing TTL-based model sets a random value to the timer T_1 only at cache miss instant t_0 as shown in Fig. 2(b).
TTL-based models are used to describe space-driven policies (like LRU as shown in Fig. 2(a), FIFO/RND as shown in Fig. 2(b) [4, 10, 13]), time-based policies (like DNS [11], modern-DNS [15]), and space-time policies (like Pra-TTL caches [6], Amazon ElastiCache and Squid web cache [3]).

2.5 Other considerations

As previous work, we are interested by the hit probability $H_{n,i}$ (resp. the occupancy $O_{n,i}$), defined as the probability



(a) Renewing TTL model



(b) Non-renewing TTL model

Figure 2: Behaviour of a file in the cache.

that file f_i is in cache v_n at request instants $\{t_k(n,i)\}_{k\geq 0}$ (resp. at any time instant t). The global (or average) hit probability $\overline{H}_n$ and the Miss Probability Ratio MPR_n between predictions of our approximation and simulation results at cache v_n are also calculated [16].
We assume that all files have the same size [16] or can be divided in small chunks of identical size [10]. Hence we express the cache size in terms of the number of files/chunks it can hold at any given moment. We also assume that files become available at cache v_n once at miss instants $\{m_{n,l}, l \geq 0\}$.

3. SINGLE CACHE FRAMEWORK

In this section, we derive new exact results on caches. We then revisit the notion of *characteristic time*, generalize the concept to caching systems described by TTL-based models, and address operations that modify request streams.

3.1 On the steady state of isolated caches

For readability, we omit the subscript n that refers to the cache label. Let $\mathcal{I}_i(t)$ be a binary random variable indicating that file f_i is in the cache at time t. Assume the limit $\mathcal{I}_i = \lim_{t\uparrow\infty} \mathcal{I}_i(t)$ exists. Since there is a copy of f_i or not in the cache at any time depending on whether $\mathcal{I}_i = 1$ or 0, the occupancy of file f_i in the cache is calculated as it follows $O_i = P(\mathcal{I}_i = 1) = E[\mathcal{I}_i]$. Note that O_i denotes the expected number of copies of file i in the cache. Let $\mathcal{I} = \sum_{i=1}^{K} \mathcal{I}_i$ be the total number of files in the cache at any time and Q_i the sojourn time of file f_i. For all caching systems, metrics of interest are related to the sojourn time of the file in the cache as follows.

LEMMA 1 (LITTLE'S LAW). *For caching systems, the metrics of interest and expected sojourn time are related by*

$$O_i = \Lambda_i(1 - H_i) \times E[Q_i], \ \forall \ f_i. \tag{1}$$

PROOF. By applying Little's law, (1) is established as follows. First, we note that the expected number of copies for file f_i in a cache is O_i. Second, the rate at which file f_i enters the cache is the miss rate $\Lambda_i \times (1 - H_i)$. Third, the expected time that file f_i spends in the cache is $E[Q_i]$. □

Unlike capacity-driven policies, the number of files cached using expiration-based policies is not bounded in principle. However, practical considerations require to not exceed a certain memory occupation level C. The following result provides conditions for the latter to hold.

LEMMA 2 (CHARACTERISTIC EQUATIONS). *For caching systems, the total number of files in the cache at steady state $\mathcal{I}$ equals the cache capacity C almost surely if and only if*

$$\sum_{i=1}^{K} O_i^p = C, \quad p = 1, 2. \tag{2}$$

PROOF. Assuming (2), we show that $E[\mathcal{I}] = C$ by noting that $\sum_{i=1}^{K} O_i = \sum_{i=1}^{K} E[\mathcal{I}_i] = E[\mathcal{I}]$. Then, $\text{Var}[\mathcal{I}] = 0$ since $\text{Var}[\mathcal{I}] = \sum_{i=1}^{K} \text{Var}[\mathcal{I}_i] = \sum_{i=1}^{K} O_i - \sum_{i=1}^{K} O_i^2$. It follows that $P\left(|\mathcal{I} - E[\mathcal{I}]| \geq \eta^2\right) \leq \frac{\text{Var}[\mathcal{I}]}{\eta^2} = 0, \ \forall \eta > 0$, thanks to Chebychev's inequality. Now if $I = C$ almost surely, then $P\left(|\mathcal{I} - C| \geq \eta\right) = 0, \forall \eta > 0$. By calculating the first two moments of I with the condition $\{|\mathcal{I} - C| \geq \eta\}$, one can easily show that (2) holds necessarily . □

REMARK 1 (CHE APPROXIMATION [4, 13]). *For capacity-driven policies (such as FIFO, LRU, RND, and variants [13]), the total number of files in the cache at steady state is constant $\mathcal{I} = C$ and $\sum_{i=1}^{K} O_i = C$ is necessary and sufficient.*

For general TTL models, there are several ways to choose per-file timer distributions $\{T_i(t) = P(T_i < t), \ \forall f_i\}$ such that (2) holds. Here are three possible choices:

- **Cache Characteristic Distribution.** Here, files have the same TTL distribution $T_i(t) = T(t), \ \forall f_i$ characterized its mean $T = E[T_i], \forall f_i$ solution of (2) with $p = 1$. Our cache characteristic distribution for general TTL-based models is consistent with *Che's approximation* [4, 10, 13] for capacity-driven policies where all files in the cache have (approximately) the same TTL distribution $T(t)$ which depends on its mean only. [13] showed ingeniously that $T(t)$ is approximately deterministic for LRU/FIFO caches (i.e. $T_i \approx E[T_i] \approx T, \forall f_i$) and exponential for RND caches (i.e. T_i is exponentially distributed with mean $E[T_i] \approx T, \forall f_i$).
- **Cache Characteristic Moments.** In this case, per-file TTL distributions are no more identical as in the previous case, but they have identical first two moments $E[T_i] = T$ and $\text{Var}[T_i] = \sigma^2, \ \forall f_i$ solutions of (2) with $p = 1$ and 2.
- **Cache Characteristic Time.** This case is stated as

ASSUMPTION 2 (CHARACTERISTIC TIME). *TTLs have the same mean $E[T_i] = T, \ \forall f_i$ and (2) holds with $p = 1$.*

Assumption 2 allows two different files, say i and j, to be cached using two different TTL distributions $T_i(t)$ and $T_j(t)$. Given that only $E[T_i] = T$ for all files f_i, per-file optimal TTL distributions can be set according to per-file request processes as proved in [9, Prop. 3.4] and [15, Prop. 4]. Assumption 2 is more general than *Che's approximation* (which is only a particular capacity-constrained TTL-based cache model with identically distributed per-file TTLs). In general, the characteristic time T in TTL-based cache models may be defined even with different per-file TTL distributions $\{T_i(t), \forall f_i\}$. Its value T is bounded as follows.

PROPOSITION 1 (BOUNDS). *Under Assumption 2, the characteristic time $T = E[T_i], \forall f_i$ is bounded by*

$$T_{\min} = \frac{C}{\Lambda} \leq \quad T \quad \leq T_{\max} = \frac{C}{\Lambda \times \overline{M}} \tag{3}$$

where $\overline{M} = \sum_{i=1}^{K} \frac{\Lambda_i(1-H_i)}{\Lambda}$ is the average miss probability and $\Lambda = \sum_{i=1}^{K} \Lambda_i$ is the total rate on the cache.

PROOF. From their definitions, Q_i and T_i are stochastically ordered as follows $Q_i \geq T_i$. By taking the expectation on the latter inequality and then replacing $E[T_i]$ by the same constant T, we obtain $E[Q_i] \geq T, \ \forall f_i$. Using the latter inequality in Lemma 1, we obtain $O_i \geq \Lambda_i(1 - H_i)T, \ \forall f_i$. Since $\Lambda_i E[T_i]$ is the expected number of copies of file f_i requested within the time interval $E[T_i] = T$ and O_i is the expected number of copies of file f_i in the cache, it follows that $O_i \leq \Lambda_i T, \ \forall f_i$. Applying (2) in Lemma 2 with $p = 1$, bounds of T are obtained as in (3). □

REMARK 2 (STATIONARY REQUESTS). *Proofs of Lemmas 1 and 2 and Proposition 1 do not require Assumption 1, but stationarity and ergodicity of request processes only.*

3.2 Metrics of interest and characteristic time

In this section, we present metrics of interest and our characteristic time approximation (CTA) on isolated caches.

We consider that request streams are described by hyper or shifted-exponential renewal processes based on the value of the square coefficient of variation (c_v^2) of inter-request times [17]. When $c_{v,i}^2 \geq 1$, $F_i(t)$ is a hyper-exponential CDF i.e. $F_i(t) = 1 - (p_1 e^{-\theta_1 t} + p_2 e^{-\theta_2 t})$, $p_1 + p_2 = 1$, $t \geq 0$, $p_1\theta_1^{-1} = p_2\theta_2^{-1}$; otherwise, it is a shifted exponential CDF i.e. $F_i(t) = 1 - e^{-\theta(t-\tau)}, \ t \geq \tau$.

It is known from [6, 15] that metrics of interest of file f_i in non-renewing (resp. renewing) TTL-based cache models are given by $H_i = \frac{E[R_i(T_i)]}{1+E[R_i(T_i)]}$ (resp. $H_i = E\left[F_i(T_i)\right]$) and $O_i = \frac{\Lambda_i E[T_i]}{1+E[R_i(T_i)]}$ (resp. $O_i = E\left[\hat{F}_i(T_i)\right]$). We derive explicit formulas for hyper/shifted-exponential renewal request processes in our technical report [7, Sect.4].

The value of T is approximated using a novel technique described in Algorithm 1. Its convergence on isolated caches is studied as follows. Thanks to Lemma 2, the characteristic time $T = E[T_i], \forall f_i$ and per-file TTLs $\{T_i, f_i \in \mathcal{F}\}$ satisfy

$$C = \sum_{i=1}^{K} \frac{\Lambda_i E[T_i]}{1 + E\left[R_i(T_i)\right]} \text{ and } C = \sum_{i=1}^{K} E\left[\hat{F}_i(T_i)\right] \tag{4}$$

for non-renewing and renewing TTL-based models respectively. Under Assumption 2, we apply Jensen's inequality and show that $\sum_{i=1}^{K} O_i \geq \sum_{i=1}^{K} \frac{\Lambda_i T}{1+R_i(T)}$ for non-renewing TTL models since $R_i(t)$ is a concave function when $F_i(t)$ is concave. It is easy to check that the hyper/shifted-exponential CDFs are indeed concave functions. Since (4) holds, it follows that $\sum_{i=1}^{K} \frac{\Lambda_i T}{1+R_i(T)} \leq C$. For renewing TTL-based models, we rely on similar arguments. Thanks to Assumption 2 and Jensen's inequality, $\sum_{i=1}^{K} O_i \leq \sum_{i=1}^{K} \hat{F}_i(T)$ since $\hat{F}_i(t)$ is a concave function. However, it follows from (4) that $\sum_{i=1}^{K} \hat{F}_i(T) \geq C$. Let us denote by $\phi(T) = C - \sum_{i=1}^{K} \frac{\Lambda_i T}{1+R_i(T)}$ (resp. $\phi(T) = C - \sum_{i=1}^{K} \hat{F}_i(T)$) in the case of non-renewing (resp. renewing) TTL based models. The function $\phi(T)$ is strictly decreasing, twice continuously differentiable, and the root is unique and simple (multiplicity is one). Thanks to Proposition 1, the zero of $\phi(T)$ belongs to $[\frac{C}{\Lambda}; \infty)$. Algorithm 1 implements the secant method to approximate the zero of $\phi(T)$ with the sequence:
$\forall I \geq 0$, $T^{(0)} = \frac{C}{\Lambda}$ and $T^{(I+1)} = T^{(I)} + \frac{T^{(I)} - 0}{\phi(T^{(I)}) - \phi(0)}\phi(T^{(I)})$.
Since $T^{(0)}$ is tight lower bound (and thus a good estimate of T) and $\phi(T)$ is strictly decreasing (and not wiggly) in $[T^{(0)}; \infty)$, it is known that the convergence of the secant method (and thus Algorithm 1) is superlinear with an order of convergence equal to the golden ratio $\frac{1+\sqrt{5}}{2}$.

REMARK 3 (QUADRATIC CONVERGENCE). *Algorithm 1 converges quadratically if Newton's method is implemented. This result holds also for general and concave CDF $F_i(t)$.*

3.3 Cache network operations

Exact characterizations of miss, split, and aggregated request processes exist, but they are too complex and computationally expensive for practical interest [3, 5, 6, 14, 15]. Hence, for each cache operation we calculate only the first two moments of inter-request times in the resulting process and we use a two-moment matching technique to fit them to hyper/shifted-exponential renewal processes [17].

3.3.1 Filtered request streams (Miss process)

For non-renewing (resp. renewing) TTL-based cache models, the two first moments of inter-miss times of file f_i are $E[Y_i] = \frac{E[X_i]}{(1+L_i^*(0))^{-1}}$ and $E[Y_i^2] = \frac{E[X_i^2]}{(1+L_i^*(0))^{-1}} - 2E[X_i](L_i^*)'(0)$
(resp. $E[Y_i] = \frac{E[X_i]}{(1-L_i^*(0))}$ and $E[Y_i^2] = \frac{E[X_i^2]}{(1-L_i^*(0))} - \frac{2E[X_i](L_i^*)'(0)}{(1-L_i^*(0))^2}$)
where explicit formulas of $L_i^*(s) = \int_0^\infty e^{-st}(1 - T_i(t))dR_i(t)$ (resp. $L_i^*(s) = \int_0^\infty e^{-st}(1 - T_i(t))dF_i(t)$.) The LSTs $L_i^*(s)$ take simple expressions when requests are described by hyper/shifted exponential renewal processes previously introduced. One can easily check that our closed-form expressions of $E[Y_i]$ and $E[Y_i^2]$ are consistent with the formulas derived by Melazzi et *al.* [14] in the specific case where TTLs are constant (and not random variables as generally assumed in this paper) and always redrawn at cache hit (and thus limited to the model described in Fig. 2(a).)

3.3.2 Split request streams (Routing departures)

In this section, we consider that a cache may route its missed requests towards J possible caches or destinations. Our aim is to describe (i.e. calculated the two first moments of) the sequence of requests that are forwarded to the j-th destination. Existing works [16, 9, 15, 3] consider that this request forwarding operation is memoryless and they model the splitting process by a Bernoulli process. However, several destinations are often allowed within more realistic networks to enable load balancing or congestion awareness based on a recent history. We therefore propose to model this request splitting operation by using a discrete time Markov chain process $\{\xi_{m_l}, l \geq 0\}$ on a state space $\{1, 2, \ldots, J\}$ such that requests occurring at miss instants $\{m_l, l \geq 0\}$ are sent to the j-th destination if $\xi_{m_l} = j$.
Further, we assume that the Markov chain $\{\xi_{m_l}, l \geq 0\}$ is lumpable and without loss of generality we characterize the sequence of requests sent to the destination labelled 1. First,

the J states of $\{\xi_{m_l}, l \geq 0\}$ are lumped into two states $I = \{1\}$ and $O = \{2, \ldots, J\}$. Second, we consider the resulting two-state Markov chain $\{\xi_{m_l} = I \text{ or } O, l \geq 0\}$ and its transition probabilities $r_{I,I} = P(\xi_{m_{l+1}} = I | \xi_{m_l} = I)$ and $r_{O,I} = P(\xi_{m_{l+1}} = I | \xi_{m_l} = O))$. Its stationary distribution is $r_I = P(\xi_{m_l} = I) = \frac{r_{O,I}}{1+r_{O,I}-r_{I,I}}$. The first two moments of inter-request times $Y_{i\to 1}$ of file f_i at destination 1 are related to those of the original process.
$E[Y_{i\to 1}] = \frac{E[Y_i]}{r_I}$ and $E\left[Y_{i\to 1}^2\right] = \frac{E[Y_i^2]}{r_I} + \frac{2(1-r_I)E[Y_i]^2}{r_I r_{O,I}}$.

3.3.3 *Aggregated request streams (Routing arrivals)*

In this section, we consider that a cache may be fed by requests originating from several sources (e.g. exogenous requests sent by its local users, and the endogenous missed requests forwarded by other caches). Our goal is to characterize the two first moments of inter-request times in the overall arrival process. It is known from [4, 6, 15, 3] that the exact characterization of this aggregated process may be significantly complex and [16, 13] showed that the Poisson assumption is source of inaccuracy. Hereafter, we describe a computationally-efficient and accurate methodology to approximate this superposed process. Without loss of generality, we consider that requests arrive at a cache from J sources labelled $j = 1, 2, \ldots, J$. Let $E[X_{i,j}]$ and $E\left[X_{i,j}^2\right]$ be the first and second moment of inter-request times of source j. If the calculation of the first moment of inter-arrival times $E[X_i]$ of requests for file f_i is straightforward $E[X_i] = \left(\sum_{j=1}^{J} E[X_{i,j}]^{-1}\right)^{-1}$, we propose a new formula to approximate the second moment $E[X_i^2]$ in the aggregated process. $E[X_i^2] = 1 + c_i^2 E[X_i]^2$ where

$$c_i^2 = w_i c_a^2 + (1 - w_i) c_p^2, \quad c_a^2 = \sum_{j=1}^{J} \frac{E[X_i]}{E[X_{i,j}]} c_{i,j}^2, \quad c_p^2 = 1, \quad (5)$$

$c_{i,j}^2 = E\left[X_{i,j}^2\right] / E[X_{i,j}]^2 - 1$, and the per-file weight w_i is

$$w_i^{-1} = 1 + \left(1 - \frac{C \times E[X_i]^{-1}}{\sum_{i=1}^{K} E[X_i]^{-1}}\right)^2 \left[\sum_{j=1}^{J} \left(\frac{E[X_i]}{E[X_{i,j}]}\right)^2\right]^{-1} \quad (6)$$

Theoretical details of our aggregated process approximation are provided in our technical report [7, Sect.5.1.3].

4. CACHE NETWORK APPROXIMATION

In this section, we extend our single cache framework to approximate performance metrics of heterogeneous cache networks with arbitrary topology. First, we define the notion of *routing topology*. We then describe our cache network algorithm (CNA) to handle general networks where bidirectional flows may occur in presence of multiple routing topologies.

4.1 Case of multiple routing topologies

The *routing topology* of a file refers to the subset of nodes that receive/exchange requests of that file. In this paper, we consider that requests of each file are routed as a *Directed Acyclic Graph* (DAG) built on top of the arbitrary network topology. In the following, the terms DAG and *routing topology* are used equivalently.

In Fig. 3 for instance, two classes $\mathcal{F}_1$ and $\mathcal{F}_2$ of content items are shown in green and blue respectively. Content items of $\mathcal{F}_1$ are stored permanently at the server connected to node

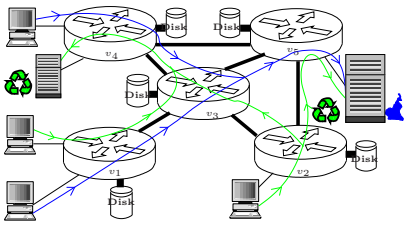


Figure 3: Two routing topologies and bidirectional flows.

v_5 and their requests are routed on the tree (displayed in blue) while those of class $\mathcal{F}_2$ are permanently available at two different servers and their requests are routed on the polytree (displayed in green). Requests of items of classes $\mathcal{F}_1$ and $\mathcal{F}_2$ are moving in opposite directions (or equivalently it exists a bidirectional flow) between nodes v_3 and v_4. Hence, states of caches 3 and 4 are dependent. Their TTLs T_3 and T_4 are coupled and solutions of a system of equations (Cf. Lemma 2). Instances of such systems of equations are available in our technical report [7, Eq.(15)] for more details.

Our algorithm to tackle this issue is described as follows.
• **Input:** network topology $\mathcal{G}(V, E)$ of size $N = |V|$, cache neighbours $\mathcal{N}(n) \subseteq V$, policies P_n and sizes C_n, file catalogue $\mathcal{F}$ of size $K = |\mathcal{F}|$, routing topologies $\{\text{DAG}_i, f_i \in \mathcal{F}\}$, exogenous requests $\{\text{rate } \lambda_{n,i} \text{ and scv } c_{n,i}^2, v_n \in V, f_i \in \mathcal{F}\}$.
• **Output:** characteristic time T_n, average miss probability $\overline{M}_n$, per-file metrics of interest {hit probability $H_{n,i}$ and occupancy $O_{n,i}$}, per-file aggregated request process {rate $\Lambda_{n,i}$ and scv $c_{X,n,i}^2$}, per-file miss request process {rate $\nu_{n,i}$ and scv $c_{Y,n,i}^2$} at each cache $v_n \in V$.
• **Procedure:** Algorithm 1 starts with an initialization step where all caches are assumed to have miss probabilities of one. The consequence is that the miss process of a node is initialized by its aggregated arrival processes. Then the characteristic times are also initialized to their lower bound. After, this initialization step, Algorithm 1 updates the miss processes of the caches using the initial value of the TTLs. Then it recalculates the aggregated streams; finally, it updates the TTL values at each cache. Algorithm 1 halts when all TTL values at all nodes of the network have converged.

4.2 Practical concerns on Algorithm 1

In this section, we show the convergence of our cache network algorithm, its polynomial-time complexity, and its properties for large scale implementations.

On the convergence. Intuitively, Algorithm 1 converges since sequences of TTL values are increasing and bounded (see Proposition 1). More formally, Algorithm 1 finds the root $\vec{T} = (T_1, \ldots, T_N)$ of the system of equations $\vec{\Phi}(\vec{\tau}) = (\Phi_1(\vec{\tau}), \ldots, \Phi_N(\vec{\tau})) = \vec{0}$, where $\vec{\Phi}(.) : \mathbb{R}^N \to \mathbb{R}^N$, $\vec{\tau} = (\tau_1, \ldots, \tau_N)$ is the characteristic time vector, and $\Phi_n(\vec{\tau}) = C_n - \sum_{i=1}^{K} \frac{\Lambda_{n,i}\tau_n}{1+R_{n,i}(\tau_n)}$ (resp. $C_n - \sum_{i=1}^{K} \hat{F}_{n,i}(\tau_n)$) for non-renewing (resp. renewing) TTL based models. Note that the dependence of $\Phi_n(\vec{\tau})$ on the characteristic times τ_m, $(m \neq n, v_m \in V)$ is strongly embedded in the aggregated request arrival process at cache v_n via the CDFs $\{F_{n,i}(.)\}$. Since $\nabla\Phi_n(\vec{\tau}) < 0$ at $\forall \vec{\tau} > 0$ and $\frac{\partial^2 \Phi_n(\vec{\tau})}{\partial \tau_n^2} > 0$, Algorithm 1 calculates updates of the characteristic time $T_n^{(I)}$ at cache

Algorithm 1: CNA on arbitrary graph $\mathcal{G}(V, E)$

$T_n^{(0)} \leftarrow 0$; $I \leftarrow 1$;
for *each node* $v_n \in V$ **do**
  Initialize per-file miss probabilities to one
  Initialize per-file miss and aggregate processes
  for *each file* $f_i \in \mathcal{F}$ *such that* $v_n \in DAG_i$ **do**
    if v_n *is an edge of* DAG_i **then**
      Set aggregate streams to exogenous process
    else
      Merge the file request streams (Sect. 3.3.3) if multiple sources: $\forall v_l \in \text{DAG}_i$ & $v_n \in \mathcal{N}(l)$
    end
    Set miss streams to aggregated processes
    Split the file miss process (Sect. 3.3.2) if multiple destinations: $\forall v_m \in \text{DAG}_i \cap \mathcal{N}(n)$
  end
  Initialize the characteristic time (Prop. 1)
    $\Lambda_n^{(1)} \leftarrow \sum_{i=1}^{K} \Lambda_{n,i}^{(1)}$; $T_n^{(1)} \leftarrow C_n/\Lambda_n^{(1)}$;
end
while $\sqrt{\frac{1}{N}\sum_{n=1}^{N}(T_n^{(I)} - T_n^{(I-1)})^2} > \epsilon$ **do**
  $I \leftarrow I + 1$;
  for *each node* $v_n \in V$ **do**
    Update per-file miss and aggregate processes
    for *each file* $f_i \in \mathcal{F}$ *such that* $v_n \in DAG_i$ **do**
      Calculate the file miss process (Sect. 3.3.1)
      Split the file miss process (Sect. 3.3.2) if multiple destinations: $\forall v_m \in \text{DAG}_i \cap \mathcal{N}(n)$
      Merge the file request streams (Sect. 3.3.3) if multiple sources: $\forall v_l \in \text{DAG}_i$ & $v_n \in \mathcal{N}(l)$
      Calculate per-file CDFs (Appendix. A) and metrics of interest (Sect. 3.2)
    end
    Update the characteristic time (Sect. 3.2)
    if *cache policy* P_n *is Non-renewing TTL* **then**
      $\bar{M}_n^{(I)} \leftarrow \sum_{i=1}^{K} \frac{\Lambda_{n,i}^{(I)}}{\Lambda_n^{(I)}}(1 + R_{n,i}(T_n^{(I-1)}))^{-1}$;
      $T_n^{(I)} \leftarrow C_n/(\Lambda_n^{(I)} \times \bar{M}_n^{(I)})$
    else
      $C_n^{(I-1)} \leftarrow \sum_{i=1}^{K} \hat{F}_{n,i}^{(I)}(T_n^{(I-1)})$;
      $T_n^{(I)} \leftarrow C_n \times T_n^{(I-1)}/C_n^{(I-1)}$
    end
  end
end

v_n by using the secant method i.e. approximating the slope with $\frac{\Phi_n(\vec{T}^{(I-1)}) - \Phi_n(\vec{0})}{T_n^{(I-1)}}$ where $\vec{T}^{(I-1)}$ is the characteristic vector at the previous iteration. Hence, Algorithm 1 converges superlinearly in each component $\Phi_n(\vec{\tau})$. The convergence is quadratic if Newton's method is used instead.

On the complexity. Instructions in bold font are basic operations of Algorithm 1. Its complexity is of order of $O(NIK)$ where I the total number of iterations is bounded by $I \leq \Delta_{\mathcal{G}}^2$ with $\Delta_{\mathcal{G}}$ the diameter of the network topology. The equality holds i.e. $I = N^2$ in the (worst) case of the tandem network of N nodes (similar to Fig. 1) with bidirectional flows through all nodes.

In the case of *unique routing topology model* which is widely studied in the literature (e.g. networks with linear [14], tree [4, 13], polytree [6, 15], feed-forward topologies [3]), Algorithm 1 needs only $I = \Delta_{\mathcal{G}}$ iterations.

On the implementation. Approximations [4, 14, 13, 16] are limited by the size and tree-based topology of networks. Unlike, Algorithm 1 scales easily as the network and catalog sizes increase. It can be implemented in distributed MapReduce jobs of *Apache Giraph* software [2]. During the initialization phase, each job associated to a node reads the network and routing configurations. It then stores labels of its neighbours and identifiers of files it may receive. During each super-step, each job executes once the while-loop of Algorithm 1 and exchanges parameters of its node request streams and characteristic time at the end. All jobs halt if the root mean squared error of TTLs is less than a given threshold.

5. EVALUATION RESULTS

In this section, we evaluate the accuracy of our model by comparing its predictions of per-file hit probabilities on networks where caches are running space-driven policies like Least Recently Used (LRU), Random replacement (RND), and First-In-First-Out (FIFO). Indeed, the existence of deterministic (resp. exponentially distributed) characteristic times for LRU and FIFO (resp. RND) policies (see [4, 10] and [13] for more details) allows us to describe them with TTL-based models as shown in Figure 2. We consider as "exact" the metrics of interest obtained via long event-driven simulations ($\approx 16.77 \times 10^6$ events generated). Files exogenous requests arrive at some nodes of the network with rate $\Lambda = 1$ and Zipf popularity of parameter $\alpha = 0.7$.

5.1 Characteristic time approximation (cta)

TTL models of isolated LRU, FIFO, and RND caches produce accurate predictions of per-file hit probabilities [4, 13, 14]. Hence, our goal here is to validate the accuracy of Algorithm 1 when approximating cache characteristic times. We consider caches of size $C = 100$, a catalogue of size $K = 10^5$, and Poisson request traffic with total request rate $\Lambda = 1$. The characteristic times or TTLs $T_{\text{POLICY}}^{\text{cta}}$ where POLICY is either LRU, RND or FIFO are calculated by solving (2) with $p = 1$ of Lemma 2. Meanwhile, their approximate values $T_{\text{POLICY}}^{(n)}$ are given by Algorithm 1. Figure 4 shows that our algorithm converges after one iteration starting from the lower bound $T^{(0)} = C/\Lambda$ to the cache characteristic times. This convergence was also observed with Interrupted Poisson Processes (IPPs) of same rate $\Lambda = 1$ and scv of inter-request times $c_v^2 = 1.5$.

Thanks to these preliminary experiments, we made two interesting observations not yet reported in the literature.

For both Poisson and IPP request models, we observed that characteristic times of FIFO and RND caches are approximatively equal. Moreover, miss streams of FIFO (resp. RND) caches fed by Poisson traffic are accurately described by shifted-exponential renewal processes.

5.2 Performance metrics on cache networks

In this section, exogenous request streams are still described by Poisson processes. Moreover, we call *Poisson approximation* the specialization of Algorithm 1 when considering the first moment of inter-request times only (and setting $w_i = 0, \forall f_i$ in (5)). Another specialization, that we

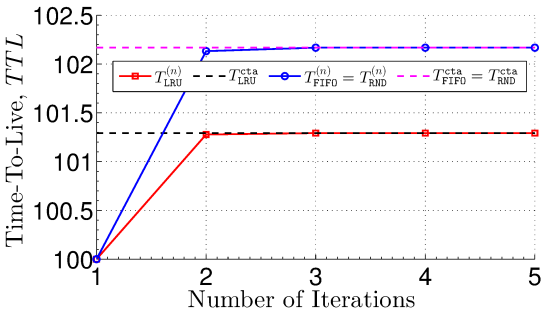


Figure 4: Approximations of TTLs $T^{\text{cta}}_{\text{POLICY}}$ for LRU, RND and FIFO policies with Algorithm 1. Poisson arrivals ($\Lambda = 1$) and Zipf popularity ($\alpha = 0.7$). $C = 100$, $K = 10^5$.

call the *Whitt approximation*, is obtained when $w_i = 1, \forall f_i$ in (5). Finally, our general methodology when $w_i \in [0, 1]$ in (5) as calculated in (6) will be the *Hybrid approximation*. These approximations were evaluated extensively under various network configurations used for a-NET model [16] and our Hybrid approximation show better accuracy than others. Due to lack of space, we only report part of results for some metrics of interest. Additional numerical results can be found in our technical report [7, Sect.6.2 & 6.3].

5.2.1 *Poisson approximation and a-NET model*

The Poisson approximation and IRM traffic model used by a-NET model [16] can be seen as assuming that all request streams are described by Poisson processes at each node of the network. We evaluate our Poisson approximation and a-NET model on linear, binary tree, and random (i.e links are drawn uniformly at random) networks of LRU caches. For these experiments, we consider that Poisson requests arrive exogenously at each cache. Due to lack of space we only report per-file hit probabilities for the random network at the node directly connected to the servers. On all simulated cache networks (Cf. [7, Sect.6.1]) and in particular for the random network (see Fig. 5), we observed that the Poisson approximation is as accurate as a-NET model [16]. Hence, only Poisson approximation is used later in this section on other network configurations for comparison purposes.

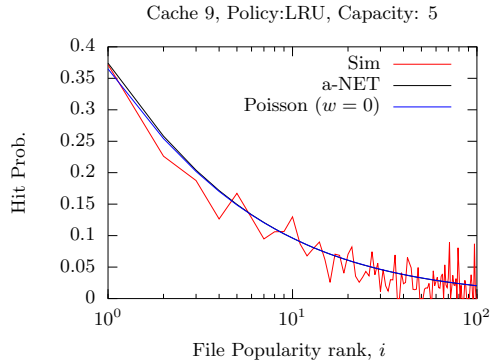
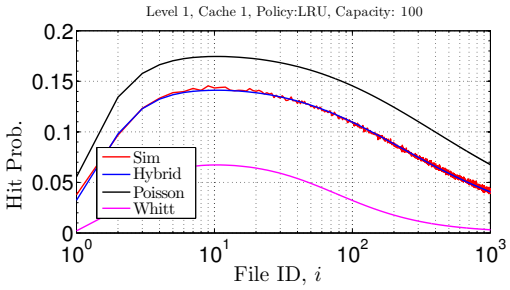


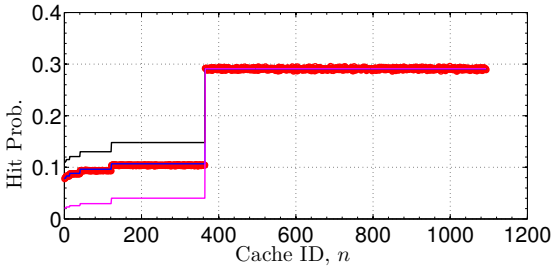
Figure 5: Simulations (Poisson arrivals $\Lambda = 1$ and Zipf popularity $\alpha = 0.7$), Poisson approximation, and a-NET model on 10 caches random network. $K = 100$, $C = 5$.

5.2.2 *Accuracy of our Hybrid approximation*

Large tree networks. We compare the Poisson, Whitt, and Hybrid approximations on homogeneous and heterogeneous tree cache networks where requests occur on leaf nodes only. In the former case, we consider a ternary tree network of depth seven having 1093 LRU caches; and, a 4-ary tree of depth five having 341 caches with capacities chosen uniformly at random within the interval $[50, 150]$ and replacement policies are selected among the FIFO, RND, and LRU policies in the latter case. The catalogue size is $K = 10^4$.



(a) Cache 1, root node



(b) Global cache hit ratio

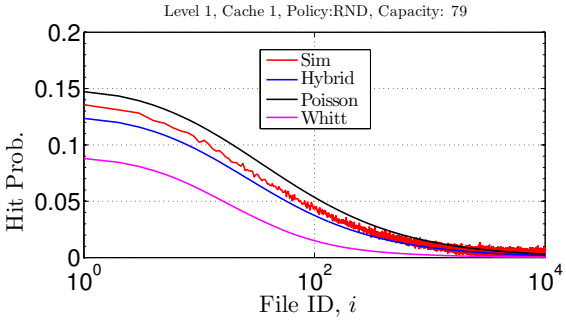
Figure 6: Simulations (Poisson arrivals on leaves and Zipf popularity). Poisson, Whitt, and Hybrid approximations. Ternary tree of LRU caches, Depth 7, $K = 10^3$, $C = 100$.

In both cases (see Figs 6 and 7), Poisson approximation overestimates the metrics of interest while Whitt approximation underestimates them. However, the Hybrid approximation outperforms others for predicting per-file hit probabilities and produces miss probability ratios close to one. Other network settings such as random networks where requests arrive on edge nodes only can be found in [7, Sect.6.2–6.3].

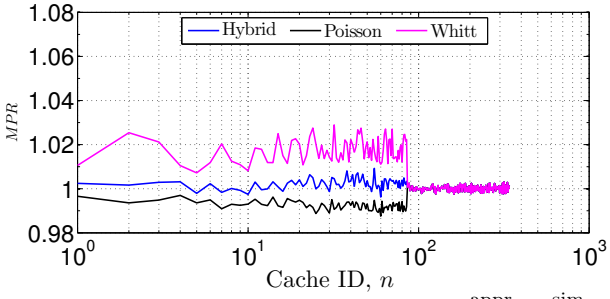
Large random networks. Finally, we consider a large random network of 100 LRU caches with capacities $C_n \in [50, 150]$ and a catalogue of size $K = 10^4$. We consider exogenous Poisson request streams at all caches. As shown in Fig. 8, our Hybrid approximation is still accurate.

6. CONCLUSION

Performance analysis of general and heterogeneous cache networks was the main goal of this work. Relying on TTL-based models, we developed an analytical framework and an accurate polynomial-time algorithm to address this problem in a quite general scope since existing models were limited either by the IRM assumption, tree topology, one-way



(a) Cache 1, root node



(b) Miss Probability Ratio, MPR $= \overline{M}_n^{\text{appr}}/\overline{M}_n^{\text{sim}}$

Figure 7: Simulations (Poisson arrivals on leaves, Zipf popularity). Poisson, Whitt, and Hybrid approximations. Heterogeneous ternary tree, depth 5, $K = 10^4, C \in [50, 150]$.

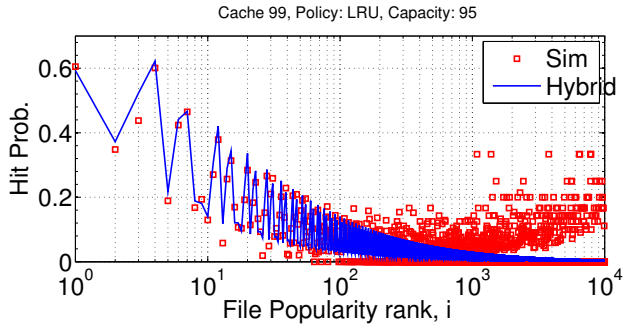


Figure 8: Hybrid approximation on a large random network of $N = 10^2$ caches, $K = 10^4$, $C \in [50, 150]$.

child-to-parent forwarding schema, homogeneous nodes, etc. Moreover, we showed the existence of *Little's Law*-like and we defined new *characteristic features* (Characteristic Equations/Moments/Distribution/Time) for caching systems behaving as generally distributed TTL-based models under stationary and ergodic request processes. Finally, simulations showed that our (Hybrid) approximation is more accurate than the existing models which addressed arbitrary cache networks since our approach captures content popularity, temporal locality, and basic cache operations on request streams in a better way by relying on hyper/shifted-exponential renewal processes. As future work, we plan to weaken our renewal assumption and to extend our framework to cache networks under correlated requests.

7. ACKNOWLEDGMENTS

The authors are deeply grateful to Christian Tanguy (Colleague at Orange Labs, Issy-Les-Moulineaux) for its comments and stimulating discussions on this work.

8. REFERENCES

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Com. Mag.*, 50(7):26–36, July 2012.

[2] Apache Foundation. Apache Giraph, http://giraph.apache.org/. Jul 2014.

[3] D. S. Berger, P. Gland, S. Singla, and F. Ciucu. Exact analysis of {TTL} cache networks. *Performance Evaluation*, 79(0):2 – 23, 2014.

[4] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: modeling, design and experimental results. *IEEE J.S.A.C*, 20(7):1305–1314, 2002.

[5] N. Choungmo Fofack. *On models for performance analysis of a core cache network and power save of a wireless access network*. PhD thesis, Feb. 2014.

[6] N. Choungmo Fofack, P. Nain, G. Neglia, and D. Towsley. Analysis of TTL-based cache networks. In *Proc. ValueTools'12*, Cargèse, France, Oct. 2012.

[7] N. E. Choungmo Fofack, D. Towsley, M. Badov, M. Dehghan, and D. L. Goeckel. An approximate analysis of heterogeneous and general cache networks. Rapport de recherche RR-8516, INRIA, Apr 2014.

[8] N. C. Fofack. Approximate models for cache analysis with correlated requests. *Actes du 10ème Atelier en Évaluation de Performances*, pages 23–24, Jun 2014.

[9] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley. Performance evaluation of hierarchical ttl-based cache networks. *Computer Networks*, 65:212–231, 2014.

[10] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for lru cache performance. In *Proc. of the 24th International Teletraffic Congress*, ITC '12, pages 1–8, 2012.

[11] J. Jung, A. W. Berger, and H. Balakrishnan. Modeling TTL-based internet caches. In *Proc. IEEE INFOCOM'03*, San Francisco, CA, USA, Mar. 2003.

[12] J. Kurose. Information-centric networking: The evolution from circuits to packets to content. *Computer Networks*, 66:112–120, 2014.

[13] V. Martina, M. Garetto, and E. Leonardi. A unified approach to the performance analysis of caching systems. In *Proc. IEEE INFOCOM*, pages 2040–2048, Apr. 2014.

[14] N. Melazzi, G. Bianchi, A. Caponi, and A. Detti. A general, tractable and accurate model for a cascade of LRU caches. *Com. Letters, IEEE*, PP(99):1–4, 2014.

[15] N. Choungmo Fofack and Sara Alouf. Modeling modern DNS caches. In *Proc. ValueTools'13*, Torino, Italy, Dec. 2013.

[16] E. Rosensweig, J. Kurose, and D. Towsley. Approximate models for general cache networks. In *Proc. Infocom'10*, San Diego, CA, USA, Mar. 2010.

[17] W. Whitt. Approximating a point process by a renewal process, I: Two basic methods. *Operations Research*, 30(1), 1982.