Analysis of Tandem PIT and CS with Non-Zero Download Delay

Huichen Dai*, Bin Liu*, Haowei Yuan†, Patrick Crowley† and Jianyuan Lu*
*Department of Computer Science and Technology, Tsinghua University
†Department of Computer Science & Engineering, Washington University in St. Louis

***Abstract*—*Collapsed forwarding* has long been used in cache systems to reduce the load on servers by aggregating requests for the same content. Named Data Networking (NDN) as a future Internet architecture incorporates this technique through a data structure called Pending Interest Table (PIT). The *request aggregation* feature suggests that PIT can be viewed as a non-reset *time-to-live (TTL) based cache*. The Content Store (CS) is a content cache placed in front of the PIT on the NDN forwarding path, so they make up a tandem cache network. To investigate the metrics of interest in this network, like the hit probability for the PIT and the CS, the expected PIT size, non-zero download delay (non-ZDD) should be taken into consideration.**

Caching policies usually assume zero download delay (ZDD), i.e., request and object arrive simultaneously, and numerous analytical methods have been proposed to study the ZDD caching policies. In this paper, after dissecting the LRU policy, we for the first time propose two LRU variants considering non-ZDD by defining separate operations for the request and object arrivals. When CS adopts the proposed LRU variants, the analysis of the CS-PIT network can still take advantage of the existing models, so the metrics of interest can be computed. Especially, the distribution for the "inter-miss" time of this network can be derived, which has not been achieved by prior works. Finally, the analytical results are verified through simulations.

I. Introduction

Named Data Networking (NDN) [1], [2] is a future Internet architecture where contents rather than hosts are first-class principals. It directly emphasizes the contents by assigning each piece of content a unique name, so that contents can be fetched, cached and redistributed by their names. Name-based requests (Interest packets) sent by end-users are interpreted by NDN routers, if not satisfied by a local cache (Content Store, CS), these requests are queried in the Pending Interest Table (PIT). PIT is a dedicated data structure used to keep track of all the currently unsatisfied Interests. If a request matches a PIT entry, it is no longer forwarded; otherwise it is recorded in PIT and forwarded according to the name-based FIB. Therefore, the NDN forwarding plane consists of three core elements – CS, PIT, and FIB. They are organized as in Fig. 1.

The PIT is a novel data structure on the NDN forwarding plane, it (i) records the Interest's path so that content (encapsulated in Data packets) can be delivered to the requesting user(s) following the reverse path; (ii) implements *collapsed forwarding* by *aggregating* the requests for the same content, providing support of native multicast[1]. Collapsed forwarding means among several continual identical requests only the first one is forwarded while the rest ones are discarded. Therefore, PIT can be viewed as an Interest cache where each entry in the PIT corresponds to a distinct Interest [3]: PIT creates a new entry for an Interest arrival, and removes this entry when the corresponding content returns; Interests that match a PIT entry will not be forwarded. Hence, the sojourn time that an Interest stays in the PIT is the download delay, making PIT conform to the concept of non-reset *TTL-based cache*.

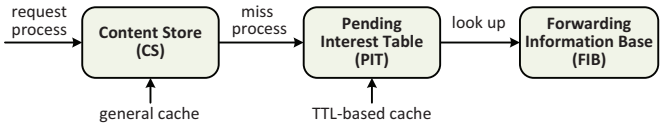


Fig. 1: Request forwarding process.

As Fig. 1 shows, CS is a local content cache located in front of PIT on the forwarding path, so they form a CS-PIT tandem cache network, where the departure request process of the CS, or the miss process[2], is the arrival request process for the PIT. The analysis of critical metrics and distributions for this network requires the consideration of non-zero download delay (non-ZDD). Prior efforts on caches often unrealistically assume that the content is downloaded instantaneously (i.e., *zero download delay*, ZDD), which violates the real world situation. A prior work [4] considers the non-ZDD when computing the expressions of the metrics, but it simply assumes that Che's approximations on *characteristic time* [5] derived from ZDD still hold for the non-ZDD case, which is however not justified theoretically.

In this paper, we assume non-ZDD modeled as a random variable or a constant, and we extend the ZDD caching policies (using LRU as an example) to the non-ZDD situation by defining separate operations for request and content arrivals. In this way, two LRU variants – LRU v-I and LRU v-II – are proposed. To the best of our knowledge, we are among the first ones to work on the non-ZDD caching policy design.

This work is sponsored by Huawei Innovation Research Program (HIRP), NSFC (61602271, 61373143, 61432009), China Postdoctoral Science Foundation (No. 2016M591182), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20130002110084), National Science Foundation (CNS-1040643, CNS-1345282).

Corresponding author: Bin Liu (liub@tsinghua.edu.cn)

[1]PIT also supports hop-by-hop forwarding, loop detection, etc.

[2]Since a cache's departure request is essentially a miss at that cache, we refer to a cache's departure request process as the "miss process", meaning that it is composed of "misses" or "missed requests".

Moreover, the analysis for the LRU variants we propose can still take advantage of the existing analytical methods originally designed for ZDD policies, including the characteristic time approximations [5].

When the CS adopts the proposed LRU variants, we model the CS-PIT network to derive the expressions for the hit probability of CS and PIT, the occupancy probability of PIT, etc., where Poisson and renewal request arrivals are considered. We also compute the inter-miss time distribution for the CS-PIT network, which is useful for studying a cache network consisting of routers. Finally, we perform extensive simulations to verify the accuracy of our models in predicting the metrics of interest.

The rest of this paper is organized as follows. §II describes the patterns and popularity law for the exogenous request process, as well as Che's approximations on cache characteristic time. §III extends existing caching policies from ZDD to non-ZDD and proposes two LRU variants. §IV analyzes the CS-PIT network mathematically to derive expressions for the metrics of interest. Extensive evaluation is conducted in §V. §VI surveys existing work on PIT implementation techniques and cache modeling. Finally, §VII concludes this paper.

II. Modeling Assumptions and Che's Approximations

In order to model the CS-PIT network, we first have some assumptions on the exogenous request process that originated directly from end-users. It is widely accepted that two main contributors to the *reference locality* are *temporal correlations* among requests and the *popularity law* of requested contents. Therefore, we describe the traffic model (reflecting the temporal correlation) followed by the popularity law. At the end of this section, we introduce the well-known Che's approximations on characteristic time for the ZDD caching policies.

A. Traffic Model

This paper uses the *renewal traffic model* to characterize the patterns of exogenous requests. In the renewal traffic model, the requests for a given object are described by an independent *renewal process*, where the inter-request times are assumed to be independent and identically distributed (i.i.d) with a general distribution. If the inter-request times are exponentially distributed, the renewal process is known as the Poisson process. This paper considers both the Poisson process and the general renewal process.

We consider a universe of N cacheable objects, labeled $i = 1, ..., N$. Let T be the random variable that describes the inter-arrival time of the requests for a specific object, then T has distribution function $F(t) = \mathbb{P}(T < t)$. The average inter-request time $\mathbb{E}[T]$ can be computed by $\mathbb{E}[T] = \int_0^\infty (1 - F(t))\,\mathrm{d}t$, so the average request arrival rate for an object is $\lambda = 1/\mathbb{E}[T]$.

Let $N(t)$ $(t \geq 0)$ denote the number of requests for an object in the interval $(0,\ t]$, so $N(t)$ is the counting process associated with the exogenous request process. Let Z_i $(i = 1, 2, \ldots)$ denote the download delay for object i, then Z_i equals the TTL assigned to a request (for object i) after it enters PIT. We assume that all $\{Z_i\}$ have the same distribution, so the subscript of Z_i can be removed. Note that in this paper we abuse the notation of a random variable for its distribution function when it causes no ambiguity, e.g., $Z(z)$ denotes the CDF for Z.

B. Popularity Law

Traffic models are commonly used in combination with an object popularity distribution, which is often referred to as *popularity law*. A popularity law determines the probability p_i that object i is requested. The most frequently observed popularity law in network traffic is a generalized *Zipf distribution*, which has been found in many types of content, such as Web pages [6], files shared using BitTorrent [7], and YouTube videos [8]. In brief, Zipf's law states that the probability of requesting the i-th most popular object is proportional to $1/i^\alpha$, where the exponent α depends on the specific system. Estimates of α in the literature for various kinds of systems range between 0.8 and 1. This paper employs the Zipf's law as the object popularity law, although our models hold in general for any given probability distribution.

C. Preliminary: Approximations on Characteristic Time

In order to tackle the modeling problem for the ZDD policies, we briefly review a simple yet powerful approach. Consider an LRU cache of size C. When an object i is evicted from this cache, the time interval between the previous request for object i and this eviction is the last time duration that object i stays in the cache. We denote this time duration by τ_i, and it is called the *characteristic time* for object i. τ_i is a random variable, it denotes the maximum sojourn time that object i can stay in the cache since the last request. Che *et al.* [5] made two approximations on τ_i:

I. τ_i itself is a constant for any given object i at an LRU cache.
II. Given any LRU cache, τ_i is a constant for all the objects in that cache, i.e., $\tau_i = \tau_j,\ 1 \leq i \neq j \leq N$.

Approximation I removes the randomness of each τ_i and Approximation II declares that all the τ_i's values are equal. These two approximations are justified theoretically [9] and experimentally [5]. They are a very powerful tool for cache performance analysis and have been used universally. This theory was originally for LRU with Poisson arrivals, but has been extended to other policies like FIFO and RND with general renewal arrivals [10]. The subscript of τ_i is omitted from now on since all τ_i are equal. The approximations reveal that the sojourn time for each object is set to the characteristic time τ when it is brought into the cache. If any cache hit happens on that object before the timer expires, the timer is reset to τ, otherwise it is evicted when the timer expires.

Computing τ is easy and requires low time complexity. Let γ_i denote the occupancy probability for object i in the cache. Define a continuous-time indicator process $I_i(t)$, which is equal to 1 when object i is present in the cache, and 0

otherwise, so $\mathbb{E}[I_i(t)] = \gamma_i$. Because the discrete expectation size that all the objects occupied in the cache equals the cache size (assuming each object has unit size), then we have $C = \sum_{i=1}^{N} I_i(t)$. Averaging both sides, we get

$$C = \sum\nolimits_{i=1}^{N} \mathbb{E}[I_i(t)] = \sum\nolimits_{i=1}^{N} \gamma_i \tag{1}$$

We can state that an object is present in the cache at time t, iff the age of the last request in the original request process for this object is no larger than τ. Let $A(t)$ denote the age of the last request for an object at time t, the limiting ($t \to \infty$) distribution function of $A(t)$ can be computed as follows [11], [12]:

$$G(a) = \mathbb{P}(A(t) \le a) = \lambda \int_0^a (1 - F(t))\,\mathrm{d}t \tag{2}$$

For Poisson request arrivals, $G(a) = 1 - e^{-\lambda a}$. Hence

$$\gamma = \mathbb{P}\{A(t) \le \tau\} = G(\tau) \tag{3}$$

Then τ can be computed by substituting (3) into (1).

Let δ denote the hit probability for an object at the LRU cache, a hit will happen iff the next request arrives before τ time elapses since the last request, which means the last inter-request time should not exceed τ, yielding

$$\delta = \mathbb{P}(T < \tau) = F(\tau) \tag{4}$$

Che's theory can be illustrated by the request process in Fig. 2. The solid circles on the top timeline shows the original requests feeding the cache, and the timeline below describes the missed requests. Upon the first cache miss, the object is brought into the cache so the following requests are hits as long as the inter-request times are smaller than τ. If no request arrives within τ time since the last hit, the object will be evicted from the cache, so the next request becomes a miss. Let R denote an object's lifespan in the cache, w denote the time interval from the object eviction to the next request, i.e., the waiting time for the next request since the eviction, and Y denote the inter-miss time, as shown in Fig 2, it is easy to see that $Y = R + w$.

The miss rate can be computed by $\lambda^m = \lambda(1 - \delta)$, so the average inter-miss time is $\mathbb{E}[Y] = 1/[\lambda(1 - \delta)]$. Define a *round* as the duration between cache's two consecutive misses, Y also denotes the length of a round.

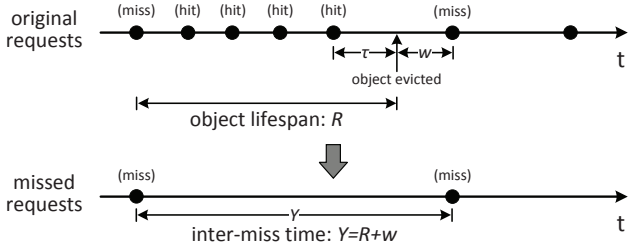


Fig. 2: Request process for Che's approximations

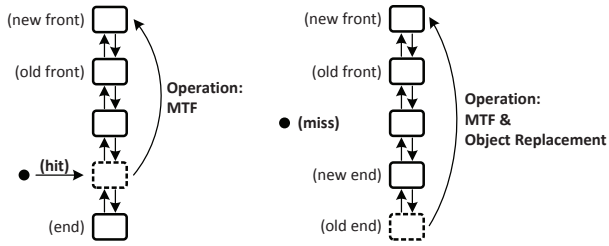
III. Caching Policy Under Non-ZDD

In this section, we first dissect the ZDD caching policy using LRU as an example, because LRU has been widely adopted, provides good performance and is reasonably simple to implement. Afterwards, we devise two elaborated variants of LRU for the non-ZDD situation, making a broad spectrum of existing analytical models [5], [13]–[16] for ZDD LRU still applicable for its non-ZDD variants. Other caching policies, such as FIFO, LFU and RND, can also be extended and analyzed for non-ZDD in the same way. Note that the LRU variants are designed for the CS in the CS-PIT network.

A. Caching Operations under ZDD

The operations for hit and miss in ZDD caching policies are defined based on the simultaneous request and content arrivals. We take LRU as an example to illustrate the operations for a ZDD policy. A typical way to implement LRU is using a doubly linked list to record the recently requested objects, where the most recently requested one is at the front of the list, while the least recently accessed one is at the end. Each list node holds the object itself as well as its name (the name is for verification to prevent lookup false positive). Fig. 3 illustrates a doubly linked list for LRU, where the cache size is 4. The operations upon a request arrival for LRU are as follows:

1. Cache hit: the matched node is moved to the front of the list, i.e., a hit triggers a Move To Front (MTF), as shown in Fig. 3(a).
2. Cache miss: the least recently accessed node (list end) is removed, and a new node is inserted to the front of the list for the newly requested object.



(a) Operation upon cache hit (b) Operation upon cache miss

Fig. 3: Operations upon requests for ZDD LRU.

For the cache miss operations, we can also interpret in this way: the list end is moved to the list front, with its object being replaced by the newly request one, meaning that a miss triggers an MTF and an object replacement, as Fig. 3(b) shows. Therefore, in summary, each request arrival in LRU triggers an MTF, regardless of a hit or a miss. The MTFs are the key operations for Che's theory, and the characteristic time is essentially the time needed for a node to move from the front to the end of the list.

While for non-ZDD LRU, there is a lag (i.e., download delay) between the request and the content, so there are two processes: the request process and the object process. Operations for request and object arrivals need to be re-defined because when a cache miss happens, an MTF is executed, but afterwards we are unable to put the object into the front node of the list because the object has not been returned yet. The operation re-definition is an open problem, which could shake the foundation of Che's approximations. Previous works [4]

barely touch this problem, but simply assume that Che's theory still holds for the non-ZDD caching policies.

Below we propose two LRU variants considering non-ZDD by designing separate operations for request and object arrivals. Both variants have no additional physical space requirement for caching real objects, neglecting the space for a hash table required by the second variant. As we shall see, Che's approximations on the characteristic time can still be applied to analyze these LRU variants. We have an assumption for the LRU variants:

- Every request will have a response.

This assumption prevents malicious requests that do not have corresponding responses from polluting the cache.

B. LRU Variant I

This LRU variant is referred to as *LRU v-I*. When a request results in a cache hit, the follow-up operations are defined as:

1. Execute an MTF for the matched node in the list;
2. If the matched node is not empty, return the object in the node to the user;
3. If the matched node is empty (refer to the operation for the missed requests below), forward this request to the upstream network, and this hit is called a *virtual hit*.

For cache miss, we define the following operations:

1. Execute an MTF for the end node;
2. Clear the object in the new front node, making it *empty*.

Intuitively, the missed request could be forwarded directly without any further operations in the non-ZDD case, and the two steps above (object eviction and MTF) should be triggered by an *object* arrival. However, as required by the logic of LRU, the list should record the reference history for the objects. As the reference history is determined by the order of requests, it is the requests' responsibility to maintain the node order in the list, leading to these operations defined for a missed *request*.

When the object returns, we let it fill its corresponding empty node, the operations are defined as:

1. Search the list (by object name) for the matched node. If the node is empty, fill the node with the object (as well as its name).
2. If the matched node has been filled, no operation is performed[3] on the list.

Therefore, in this LRU variant an MTF is always executed for a request regardless of hit or miss, which resembles the ZDD LRU. Compared with the ZDD LRU, the operations for an object are essentially a *delayed caching* of the object, because we need to wait for the object to be downloaded. During the download delay, a request's matched node cannot serve the content, which is the root reason of virtual hits.

In summary, in LRU v-I: 1) MTFs are only triggered by requests, 2) Only MTF changes the list order. *The MTFs for LRU v-I are exactly the same as the ZDD LRU, therefore Che's approximations on characteristic time remain valid for LRU v-I.* The characteristic time can still be solved via Eq. (1). Recall that R denotes the lifespan of an object in the cache for ZDD LRU, the lifespan is shortened to $R - Z$ for LRU v-I because of the download delay.

[3]This case happens for the virtual hits. When the object requested by a virtual hit returns, the matched node has been filled by the object requested by the previous missed request.

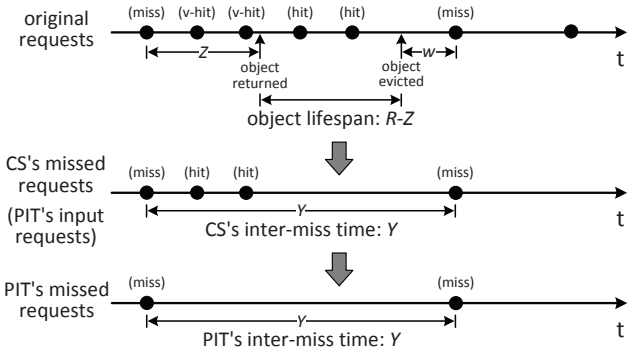


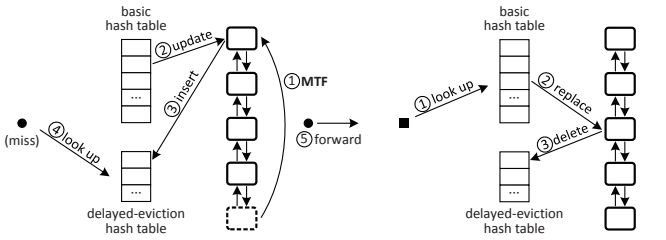
Fig. 4: Request process for LRU v-I. (*v-hit* denotes a virtual hit.)

Fig. 4 illustrates the request processes for LRU v-I. The solid circles on the first timeline represent the exogenous requests arriving at the CS. The first request results in a CS miss, and the following $N(Z)$ requests are virtual hits, as labeled by the first timeline. An amount of requests after the virtual hits are real hitting ones on the CS. Except these real hitting requests, the rest ones are CS's missed requests, which are illustrated by the solid circles on the second timeline. They continue to feed the PIT. The PIT will allocate a new entry for the first request it receives and then forward it; the CS's virtual-hit requests will hit this entry, thus are no longer forwarded. Finally, only the first miss passes through the CS-PIT network, followed by the next miss of the next round, as described by the circles on the last timeline in Fig. 4.

By comparing CS's miss process in Fig. 2 and CS-PIT's miss process in Fig. 4 (the bottom timelines in both figures), we have the following important observation:

- For the single ZDD CS that adopts LRU, and the non-ZDD CS-PIT network whose CS adopts LRU v-I, they have the *same miss process*, given the same request pattern and the same CS configuration.

The reasons are: (i) the first request passes both the ZDD CS and the non-ZDD CS-PIT network, but it gets recorded by the PIT in the latter case; (ii) the follow-up requests are satisfied by the single ZDD CS, while $N(Z)$ requests pass through the CS (virtual hits) in the CS-PIT network during the download delay, but they will hit the PIT entry, and are no longer forwarded (request aggregation at PIT), thus they cannot pass the CS-PIT network; (iii) the next miss for the single ZDD CS appears when an inter-arrival time of the exogenous requests exceeds the characteristic time. Because the CS in the non-ZDD CS-PIT network has equal characteristic time as the single ZDD CS, the same miss will emerge at the same epoch and pass through the CS-PIT network. Consequently, both miss processes are the same. And among the metrics and distributions for the non-ZDD CS-PIT network, some can be derived through analyzing the single ZDD CS.



(a) Operations upon cache miss (b) Operations upon object return

Fig. 5: Operations for cache miss and object arrival for LRU v-II.

C. LRU Variant II

A cache miss in LRU v-I evicts the object in the *victim node*[4] immediately and waits for the new object to be downloaded to fill the victim node. During the downloading time, the node is empty and unable to serve requests, which is a kind of resource waste. Here we further propose *delayed eviction* to make use of this duration: the *victim object* (assume it is object i) still remains in the victim node until the newly requested object (assume it is object j) returns, afterwards object i is replaced by object j. During the downloading time for object j, requests for object i can still be satisfied by the cache, increasing its hit probability. In this way, we have another LRU variant called *LRU v-II*.

A hash table is usually used to quickly search the objects (by name) in the linked list to avoid a linear traversal, and we call it the *basic hash table* (not shown in Fig. 3). To implement delayed eviction, we make use of another separate hash table to index the victim objects, which is referred to as the *delayed-eviction hash table*. Both hash tables are shown in Fig. 5.

The detailed operations for this LRU variant are as follows. When a request for object i arrives, it examines the basic hash table to search the cache. On a cache hit:

1. Execute an MTF for the matched node in the list;
2. Examine the name in the request against the object name in the matched node. If equal, return the object;
3. Otherwise, the object in the node is not the one requested, but a victim object waiting for the delayed eviction. Hence, this request results in a virtual hit, and we still forward it.

On a cache miss: (The steps are illustrated in Fig. 5(a).)

1. Execute an MTF operation for the list end (step ①), and update the index in the basic hash table for this node (step ②), the hash key is the name of the requested object. *Note that the object in the victim node is NOT cleared now;*
2. Insert the victim node into the delayed-eviction hash table with the hash key being the name of the victim object (step ③). Hence after this step both the basic hash table and the delayed-eviction hash table have an entry pointing to this victim node, with different keys;
3. Look up the delayed-eviction hash table for the requested object (step ④). If hit, return the object (in this case, this "missed" request eventually get satisfied thanks to the delayed eviction);
4. Otherwise forward this request (step ⑤).

When an object returns:

1. Search the basic hash table by object name to find a matched node, then replace the object in that node (meaning that the victim object is eventually evicted after the download delay);
2. Remove the index for the victim object in the delayed-eviction hash table.

This procedure is shown in Fig. 5(b). Note that the insertion and deletion operations on the basic hash table (for cache replacement) are omitted because they are trivial. *Just like the LRU v-I, the MTFs for LRU v-II are also exactly the same as the ZDD LRU, therefore Che's theory remains valid for LRU v-II with equal characteristic time.*

[4]Victim node holds the the object that is going to be replaced, or *victim object*. For LRU, the victim node is always the end node of the list.

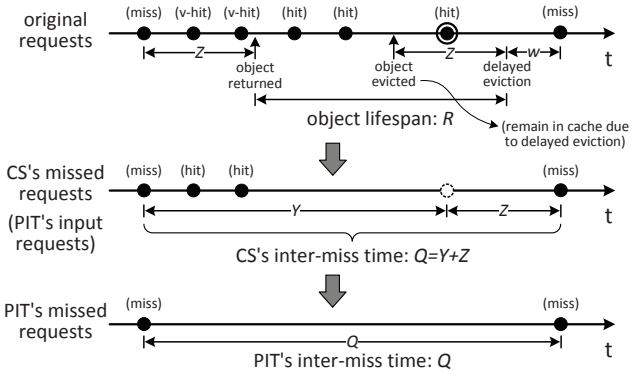


Fig. 6: Request process for LRU v-II. (*v-hit* denotes a virtual hit.)

Essentially, LRU v-II aims to extend an object's lifespan in the cache on the basis of LRU v-I. Assume object i is going to be replaced by object j due to cache miss for object j, instead of evicting object i immediately after the miss as LRU v-I does, LRU v-II let object i remain in the cache until object j returns to replace it. In this way, object i's lifespan is extended by Z (the download delay).

Fig. 6 shows the request processes for the CS-PIT network as Fig. 4 does. The top timeline in Fig. 6 shows that the real object eviction is postponed by a duration of Z compared with Fig. 4. The highlighted request (by an extra circle) in Fig. 6 results in a hit due to the delayed eviction, while the corresponding request in Fig. 4 leads to a miss. In LRU v-I, the object lifespan is $R-Z$, now the object lifespan is lengthened to $R\ (=(R-Z)+Z)$, as shown by the top timeline in Fig. 6.

Similar to LRU v-I, the first request and the following $N(Z)$ ones pass through the CS and continue to feed PIT. Afterwards, the first one is still forwarded by PIT, but the rest ones hit PIT and are discarded. Let Q denote the PIT's inter-miss time, still let w represent the waiting time after an object eviction to the next request. As described by Fig. 6, $Q=Z+R+w$, because $R+w=Y$ (denoted in Fig. 2), we have $Q=Y+Z$, meaning that the inter-miss time of LRU v-II is also lengthened by Z.

IV. Modeling the Non-ZDD CS-PIT Network

This section presents analytical models for the CS-PIT tandem network. Before elaborating on the models, we identify that there are two types of TTL-based caches. *Type-I* (non-reset): when an uncached item is brought into the cache due to a miss, a timer (with initial value of the TTL) is fired; then the item is evicted when the timer expires (regardless of any hits during the TTL). This type can be found in DNS caches [17]. As a PIT entry has a lifespan of download delay regardless if any request hits it, PIT belongs to this type.

Type-II (reset): when an uncached item is brought into the cache, a timer (initialized to the TTL) is fired, and will be *reset* to TTL upon every future cache hit for that item. According to Che's theory, an object enters the ZDD CS will have its TTL set to the characteristic time; on each hit the TTL is reset to the characteristic time. Therefore, the ZDD CS belongs to the type-II, and the TTL is the deterministic characteristic time.

A. The Hit and Occupancy Probability of CS and PIT

The expected number of renewals for the time duration $(0, t]$ is described by the *renewal function* $m(t) = \mathbb{E}[N(t)]$, and $m(t)$ satisfies the following *renewal equation* [11]

$$m(t) = F(t) + \int_0^t m(t-x)\,\mathrm{d}F(x) \tag{5}$$

Now we introduce a notation for the Laplace transform (LT). For any function $g(x)$ $(x > 0)$, we use a tilde to denote its Laplace transform, i.e.,

$$\tilde{g}(s) = \mathcal{L}\{g(x)\}(s) = \int_0^\infty e^{-sx} g(x)\,\mathrm{d}x$$

Taking LT on both sides of Eq. (5), we have

$$\tilde{m}(s) = \frac{\tilde{F}(s)}{1 - s\tilde{F}(s)} \tag{6}$$

where $\tilde{F}(s)$ is the LT of $F(t)$. Then $m(t)$ can be derived by the inverse LT of (6), so we have (see Eq. (3.9) in [18])

$$m(t) = \frac{t}{\mathbb{E}[T]} + \frac{\mathbb{E}[T^2]}{2\mathbb{E}[T]^2} - 1 + \varepsilon(t), \quad t > 0 \tag{7}$$

where $\lim_{t\to\infty} \varepsilon(t) = 0$. For Poisson process, it's known that

$$m(t) = \lambda t, \quad t > 0 \tag{8}$$

• LRU v-I:

When the CS adopts LRU v-I, for a specific object, its overall hit probability δ in the CS-PIT network equals its hit probability of the ZDD CS that adopts LRU, thus can be derived by Eq. (4). For Poisson arrivals, $\delta = 1 - e^{-\lambda\tau}$.

As mentioned, the exogenous request process can be divided into consecutive *rounds* separated by misses of the CS-PIT network, e.g., interval $[0, Y]$ is a round. Note that these rounds are statistically the same. Without loss of generality, consider the cycle starting at $t = 0$. Fig. 4 denotes that, during time $(0, Y]$, the requests in interval $(0, Z]$ pass through the CS to feed PIT and will hit an entry in the PIT, while those in interval $(Z, Y]$ are satisfied by CS. The expected number of requests within this round equals $1 + \mathbb{E}[m(Y)]$, where the expectation is computed with respect to the distribution of Y. From these requests, $\mathbb{E}[m(Z)]$ on average pass through the CS and result in PIT hits. We can now write the PIT hit probability for the given object as the expected number of hit requests divided by the expected total number of requests in a round:

$$\rho_I = \frac{\mathbb{E}[m(Z)]}{1 + \mathbb{E}[m(Y)]} = \frac{\int_0^\infty m(z)\mathrm{d}Z(z)}{1 + \int_0^\infty m(y)\mathrm{d}Y(y)} \tag{9}$$

Eq. (9) has also been proven in [17], [19]. *We defer the the computation of $Y(t)$ to the next subsection.*

The PIT occupancy probability for a given object can be expressed as the expected time spent in PIT over a round:

$$\theta_I = \mathbb{E}[Z]/\mathbb{E}[Y] \tag{10}$$

Summing up all the θ_I for all the objects can derive the expected PIT size (in terms of the number of entries).

The hit probability for the CS in the CS-PIT network can be easily derived by

$$\delta_I = \delta - \rho_I \tag{11}$$

And its occupancy probability equals the single ZDD CS, hence can be derived via (3).

• LRU v-II:

The round for LRU v-II has time length Q. By the same rationale for LRU v-I, the PIT hit probability for LRU v-II is

$$\rho_{II} = \frac{\mathbb{E}[m(Z)])}{1 + \mathbb{E}[m(Q)]} = \frac{\int_0^\infty m(z)\mathrm{d}Z(z)}{1 + \int_0^\infty m(q)\mathrm{d}Q(q)} \tag{12}$$

The PIT occupancy probability is

$$\theta_{II} = \mathbb{E}[Z]/\mathbb{E}[Q] = \mathbb{E}[Z]/(\mathbb{E}[Y] + \mathbb{E}[Z]) \tag{13}$$

The miss rate of the CS-PIT network is $1/\mathbb{E}[Q]$, so the overall hit probability for an object is

$$\delta = \frac{\lambda - 1/\mathbb{E}[Q]}{\lambda} = 1 - \frac{1}{\lambda\,\mathbb{E}[Q]} \tag{14}$$

Then the CS hit probability in the CS-PIT network is

$$\delta_{II} = \delta - \rho_{II} \tag{15}$$

The CS occupancy probability can still be computed by (3).

B. CS-PIT's Inter-Miss Time Distribution

The distribution of Y is crucial to compute the values of the metrics above for the CS-PIT network. Because the non-ZDD CS-PIT network that adopts LRU v-I has the same miss process with the single ZDD CS, we can study the single ZDD CS instead to derive their inter-miss time distribution $Y(t)$. To achieve this, we first borrow a model for general type-II TTL-based caches with renewal request arrivals [14] (denoted in Eq. (17)). Because the ZDD CS is essentially a type-II TTL-based cache with *deterministic* TTL, it is a special case for this model. Afterwards, we will provide our own method to solve the model.

For a general type-II TTL-based cache, assume that the TTL D (with $D(d)$ representing its CDF) and inter-request time T are independently distributed. Define $L(t) = \mathbb{P}(T < t,\ T < D)$ as the stationary probability that the inter-request time for

an object is smaller than both t and the TTL. Because arrivals and TTLs are independent, we have

$$L(t) = \mathbb{P}(T < t)\big(1 - \mathbb{P}(D < T)\big) = \int_0^t (1 - D(d))\,\mathrm{d}F(d) \quad (16)$$

Let $Y(t)$ denote the CDF of Y. Here is the borrowed model: Fofack *et al.* [14] have pointed out that the miss process of a single cache fed by renewal requests is also a renewal process, and $Y(t)$ satisfies

$$Y(t) = F(t) - L(t) + \int_0^t Y(t-x)\,\mathrm{d}L(x) \quad (17)$$

Because the CS that adopts the ZDD LRU is essentially a type-II TTL-based cache with deterministic TTL, i.e., $D = \tau$, the CDF for D is:

$$D(d) = \begin{cases} 0, & d < \tau \\ 1, & d > \tau \end{cases}$$

hence (16) is simplified to

$$L(t) = \begin{cases} F(t), & t < \tau \\ F(\tau), & t > \tau \end{cases}$$

then (17) becomes

$$Y(t) = \mathbf{1}_{t>\tau}\big(F(t) - F(\tau) + \int_0^\tau Y(t-x)\,\mathrm{d}F(x)\big) \quad (18)$$

Directly solving (17) for $Y(t)$ is prohibitively complicated. Usually we take LT on both sides to get $\tilde{Y}(s)$, then $Y(t)$ can be obtained by taking the inverse LT of $\tilde{Y}(s)$. Unfortunately, it is not straightforward to compute the Laplace transform of the integral term $\int_0^\tau Y(t-x)\,\mathrm{d}F(x)$ because its upper limit is not t but rather a constant τ. Next, we provide our own method to solve (18) for $Y(t)$. Let

$$H : t \mapsto F(\min(t, \tau))$$

then $\mathrm{d}H(t) = \mathbf{1}_{t\leqslant\tau}\,\mathrm{d}F(t)$, hence (18) reads

$$Y(t) = \mathbf{1}_{t>\tau}(F(t) - F(\tau) + \int_0^t Y(t-x)\,\mathrm{d}H(x)) \quad (19)$$

Let $V(t) = Y(t+\tau)$ and $K(t) = F(t+\tau)$, then (19) becomes

$$V(t) = K(t) - F(\tau) + \int_0^t V(t-x)\,\mathrm{d}H(x), \quad t > 0 \quad (20)$$

Taking Laplace transform on both sides, it follows that

$$\tilde{V}(s) = \frac{\tilde{K}(s) - F(\tau)/s}{1 - \tilde{h}(s)} \quad (21)$$

where $\tilde{h}(s) = \int_0^\infty \mathrm{e}^{-st}\,\mathrm{d}H(t) = \int_0^\tau \mathrm{e}^{-st}\,\mathrm{d}F(t)$. Since $V(t) = Y(t+\tau)$, then $Y(t) = V(t-\tau)\,u(t-\tau)$, so

$$\tilde{Y}(s) = e^{-\tau s}\,\tilde{V}(s) = e^{-\tau s}\,\frac{\tilde{K}(s) - F(\tau)/s}{1 - \tilde{h}(s)} \quad (22)$$

Afterwards, $Y(t)$ can be derived by the inverse LT of $\tilde{Y}(s)$.

If the CS in the CS-PIT network adopts LRU v-I, then $Y(t)$ is the inter-miss time distribution for the CS-PIT network. When the CS adopts LRU v-II, the inter-miss time $Q = Y + Z$. Because Y and Z are independent, the CDF for Q is the convolution of $Y(t)$ and $Z(t)$, i.e.,

$$Q(t) = Y(t) * Z(t) \quad (23)$$

If Z is deterministic, then $Q(t) = Y(t - Z)$.

Eq. (22) holds with renewal arrivals. For Poisson request arrivals, $Y(t)$ could be accurately approximated by [5]

$$Y(t) = \begin{cases} 1 - e^{-\beta(t-\tau)}, & \tau < t < \infty \\ 0, & t < \tau \end{cases} \quad (24)$$

where $\beta = 1/(\lambda^{-1}e^{\lambda\tau} - \tau)$.

V. Evaluation

In this section, we evaluate the accuracy of our models by comparing against numerical simulations. We simulate LRU, LRU v-I and LRU v-II for CS with Poisson and renewal request arrivals. The inter-request time distribution for renewal arrivals is Erlang-2. The CS size is $C = 1,000$, and requests arrive for $N = 2,000$ objects. The aggregate arrival rate for all the objects is assumed to be 5,000 per second. Object popularities follow a Zipf distribution with parameter $\alpha = 0.8$. The download delay is assumed to be exponentially distributed, when evaluating the accuracy of the models for individual objects, the average is set to 200 ms. To study the dynamics of metrics under different download delays, we also vary the average download delay from 0 to 300 ms.

A. CS-PIT Hit Probability

Fig. 7 shows the hit probability for each individual object for the CS-PIT network under LRU v-I and LRU v-II. The results are derived by modeling and simulation, and the simulation results accurately fit the modeling results with small relative error. Both subplots reveal that Poisson arrivals have higher CS-PIT hit ratio than Erlang-2 arrivals, the reason is Poisson arrivals has stronger locality of reference than Erlang-2 arrivals; and a comparison between these two subplots indicates that LRU v-II results in higher CS-PIT hit probability than LRU v-I because of the delayed eviction.

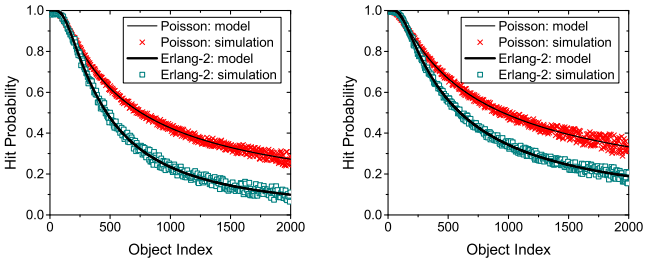
B. CS Hit Probability

Fig. 8 shows the CS hit probability of the individual objects under LRU v-I and LRU v-II. Again, Poisson arrivals results in higher CS hit ratio than Erlang-2 arrivals, and LRU v-II improves the CS hit probability due to the delayed eviction.

Fig. 9 presents the CS's overall hit probability of all the objects with increasing download delay, derived from model and simulation. Note that the results for 0 download delay in this figure correspond to the ZDD LRU. As one can guess, the non-zero download delay would degrade the overall CS hit probability, but Fig. 9 suggests that the increased download delay have minor impacts on CS's overall hit probability when utilizing LRU v-I and v-II. Therefore, LRU v-I and v-II are well-designed non-ZDD caching policies.

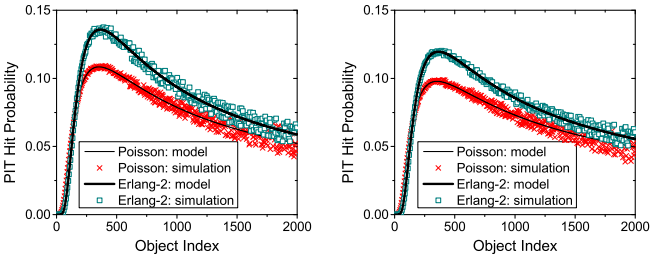
C. PIT Hit Probability

Fig. 10 shows the PIT hit probability for each individual object. Erlang-2 arrivals result in higher PIT hit probability than the Poisson arrivals, the reason is Erlang-2 arrivals lead to



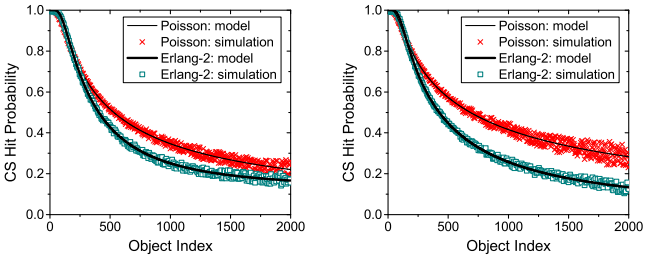
(a) LRU v-I. (b) LRU v-II.

Fig. 7: Hit prob. of individual objects for the CS-PIT network.



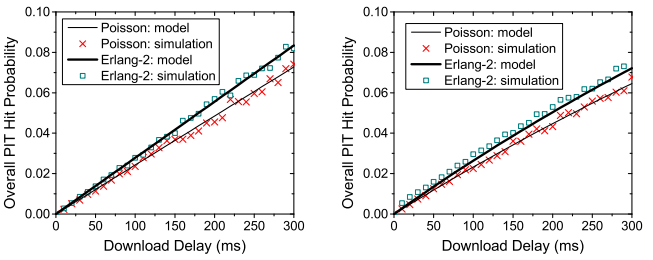
(a) LRU v-I. (b) LRU v-II.

Fig. 10: PIT hit probability for individual objects.



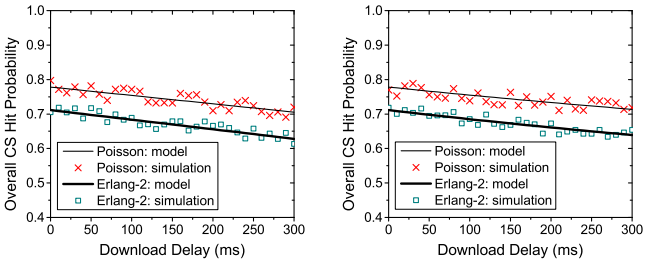
(a) LRU v-I. (b) LRU v-II.

Fig. 8: CS hit probability for individual objects.



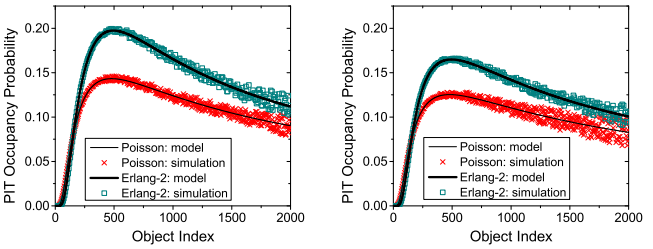
(a) LRU v-I. (b) LRU v-II.

Fig. 11: Overall PIT hit probability.



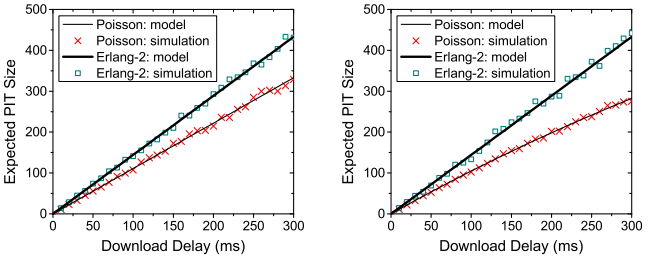
(a) LRU v-I. (b) LRU v-II.

Fig. 9: Overall CS hit probability with increasing download delay.



(a) LRU v-I. (b) LRU v-II.

Fig. 12: PIT occupancy probability for individual objects.



(a) LRU v-I. (b) LRU v-II.

Fig. 13: Expected PIT size.

lower CS hit probability, so more requests can pass through CS and arrive at PIT, leading to higher PIT hit ratio. Comparing the CS and PIT hit ratios, it is clear that the PIT contributes much fewer request hits than the CS does. Fig. 11 shows that the overall PIT hit probability grows as the download delay increases. As we can expect, the PIT hit probability is very sensitive to the download delay values. Today the network delays are mostly less than 200 ms, so the overall PIT hit ratio would be fair. Therefore, maybe the more important role of PIT is to record the Interest path.

D. PIT Occupancy Probability and Expected PIT Size

Fig. 12 presents the PIT occupancy probability for each object, i.e., the probability of having an entry for an object in the PIT, which turns out to be higher than the PIT hit ratio. The expected PIT size can be calculated by summing up the occupancy probabilities of all the objects, and the results with different download delay values are shown in Fig. 13. Because the download delay is the sojourn time that an PIT entry stays in the PIT, as one can expect that the average PIT size increases as the download delay grows.

VI. Related Work

Two critical aspects about PIT are investigated in this section. (i) system-oriented work: compact and scalable data structures for wire-speed PIT implementation; (ii) modeling work: the PIT size evaluation. Dai *et al.* [20] estimate the size by analyzing a real IP trace, and also propose the

Name Component Encoding method to accelerate PIT query. DiPIT [21] proposes using Bloom filters to implement PIT, where each interface is assigned a Bloom filter to record the incoming Interests via that interface. Yuan *et al.* [22] provide an approximate quantification of PIT size by analyzing link speeds, and propose a novel Pending Interest Table design that guarantees packet delivery with a compact and approximate storage representation. Performance comparisons between encoding methods and hash-based methods can be found in [23], and the authors recommend using both *linear-chaining* hash table and *open-addressed d-left* hash table to implement PIT.

Because PIT can be viewed as a TTL-based cache and the CS is essentially a content cache, we also survey related work on cache modeling in this section. An abundant literature exists on the performance (e.g. hit probability, search cost) of a single cache running the FIFO, RND (RANDOM), LRU replacement policy, etc. With few exceptions, exact models of caches in isolation are computationally intractable, resulting in the reliance on approximations [5], [24]. In recent years, several proposed approximate models are able to provide very accurate analytical results, e.g., Che *et al.* [5] offered two pioneering approximations on characteristic time to derive the hit probability and the inter-miss time distribution for LRU caches fed by Poisson traffic. Martina *et al.* [10] extended the approximations to other caching policies and cache networks fed by renewal traffic, but the inter-miss time distribution is untouched in that work. Networks of caches are significantly more difficult to analyze and exact solutions are hard to obtain for even two LRU (or FIFO, RND) caches in tandem. In the past, approximate models have been proposed for star and tandem networks of LRU and RND caches by [5], [9], [13]. However, these models suffer from inaccuracies as reported in [13] where the relative error could reach 16%. Recently, Fofack *et al.* [14]–[16] offered models for TTL-based isolated caches and cache networks with relative errors less than 1%. But they assume zero download delay, which would make the derived distribution for the miss stream inaccurate for the real world. However, with the LRU variants proposed in this paper, we can leverage the models in these works to analyze the CS-PIT network.

Dehghan *et al.* [4] also study the CS-PIT network. For ZDD, a cache can be modeled as a TTL reset or non-reseet cache (for LRU and FIFO) with constant timers according to Che's approximations on characteristic time. They simply assume that this conclusion drawn from the ZDD still holds for the non-ZDD, so they regard a non-ZDD CS as a TTL reset/non-reset cache with a positive downloading delay. However, this strong assumption is not theoretically justified in their work. Carofiglio *et al.* [25] focus on PIT dimensioning by a fluid model, but they do not take other metrics like PIT hit probability, inter-miss time distribution into consideration.

VII. Conclusion

The CS and PIT make up a tandem network on the NDN forwarding plane, their hit and occupancy probabilities are critical metrics for evaluation. In this paper, we for the first time propose two LRU variants that take non-ZDD into consideration. The analysis of these two LRU variants can still make use of existing analytical methods for ZDD caching policies, including the approximations on characteristic time. Analytical models for the CS-PIT network are proposed afterwards, so that the expressions for metrics of interest, as well as the distribution for the miss process, are derived. Finally, the analytical results are verified through extensive simulations.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *Proc. of CoNEXT*, 2009.

[2] L. Zhang, D. Estrin, V. Jacobson, and B. Zhang, "Named Data Networking (NDN) Project," in *Technical Report, NDN-0001*, 2010.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom'00. New York, NY, USA: ACM, 2000, pp. 56–67.

[4] M. Dehghan, B. Jiang, A. Dabirmoghaddam, and D. Towsley, "On the analysis of caches with pending interest tables," in *Proceedings of the International Conference on Information-Centric Networking*, ser. ICN'15, 2015, pp. 69–78.

[5] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, no. 7, pp. 1305–1314, 2002.

[6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proceedings of IEEE INFOCOM'99*, 1999, pp. 126–134.

[7] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *IEEE INFOCOM Computer Communications Workshops (INFOCOM WKSHPS)*, 2012, pp. 310–315.

[8] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of ACM SIGCOMM conference on Internet measurement (IMC)*, 2007, pp. 15–28.

[9] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for lru cache performance," in *Proceedings of the International Teletraffic Congress*. International Teletraffic Congress, 2012, p. 8.

[10] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *Proceedings of IEEE INFOCOM'14*, April 2014, pp. 2040–2048.

[11] L. Breuer, *Introduction to Stochastic Processes*. [Online]. Available: http://www.kent.ac.uk/smsas/personal/lb209/files/sp07.pdf

[12] S. Karlin and H. M. Taylor, "A first course in stochastic processes, 2nd edition," *Academic Press*, 1975.

[13] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *Proceedings of IEEE INFOCOM'10*. IEEE, 2010, pp. 1–9.

[14] N. Choungmo Fofack, P. Nain, G. Neglia, and D. Towsley, "Performance evaluation of hierarchical TTL-based cache networks," *Computer Networks*, vol. 65, pp. 212–231, 2014.

[15] N. C. Fofack, M. Dehghan, D. Towsley, M. Badov, and D. L. Goeckel, "On the performance of general cache networks," in *Proceedings of the International Conference on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS'14, ICST, Brussels, Belgium, Belgium, 2014, pp. 106–113.

[16] N. C. Fofack, D. Towsley, M. Badov, M. Dehghan, and D. L. Goeckel, "An approximate analysis of heterogeneous and general cache networks," *[Research Report] RR-8516, hal-00975339*, p. 36, 2014.

[17] J. Jung, A. W. Berger, and H. Balakrishnan, "Modeling TTL-based internet caches," in *IEEE INFOCOM'03*, 2003, pp. 417–426.

[18] R. G. Gallager, *Discrete Stochastic Processes*. Kluwer, 1996.

[19] O. Bahat and A. M. Makowski, "Measuring consistency in ttl-based caches," *Perform. Eval.*, vol. 62, no. 1-4, pp. 439–455, Oct. 2005.

[20] H. Dai, B. Liu, Y. Chen, and Y. Wang, "On pending interest table in named data networking," in *Proceedings of ACM/IEEE ANCS'12*, Oct 2012, pp. 211–222.

[21] W. You, B. Mathieu, P. Truong, J.-F. Peltier, and G. Simon, "Dipit: a distributed bloom-filter based pit table for ccn nodes," in *IEEE ICCCN*, 2012, pp. 1–7.

[22] H. Yuan and P. Crowley, "Scalable Pending Interest Table Design: From Principles to Practice," in *Proc. of the IEEE Conference on Computer Communications (INFOCOM'14)*, April 2014, pp. 2049–2057.

[23] M. Varvello, D. Perino, and L. Linguaglossa, "On the design and implementation of a wire-speed pending interest table," in *International Workshop on Emerging Design Choices in Name-Oriented Networking (INFOCOM WKSHPS)*, 2013, pp. 369–374.

[24] A. Dan and D. Towsley, *An approximate analysis of the LRU and FIFO buffer replacement schemes*. ACM, 1990, vol. 18, no. 1.

[25] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Pending interest table sizing in named data networking," in *Proceedings of the International Conference on Information-Centric Networking*, ser. ICN'15, 2015, pp. 49–58.