On the Analysis of Caches with Pending Interest Tables

Mostafa Dehghan[1], Bo Jiang[1], Ali Dabirmoghaddam[2], and Don Towsley[1]
[1]College of Information and Computer Sciences, University of Massachusetts, Amherst, MA
[2]Computer Engineering Department, University of California, Santa Cruz, CA
{mdehghan, bjiang, towsley}@cs.umass.edu, alid@soe.ucsc.edu

ABSTRACT

Collapsed forwarding has been used in cache systems to reduce the load on servers by aggregating requests for the same content. This technique has made its way into design proposals for the future Internet architecture through a data structure called Pending Interest Table (PIT). A PIT keeps track of *interest* packets that are received at a cache-router until they are responded to. PITs are considered useful for a variety of reasons *e.g.*, communicating without the knowledge of source and destination, reducing bandwidth usage, better security, etc. Due to the high access frequency to the PIT, it is essential to understand its behavior, and the effect it has on cache performance. In this paper, we consider a TTL-based cache with a Pending Interest Table, and analyze the cache hit probability, mean response time perceived by the users, and size of the PIT, among other metrics of interest. In our analysis, we account for the time it takes for the cache to download a file from the server defined as a random variable. We apply our model to analyze traditional caching policies LRU, FIFO, and RANDOM, and verify the accuracy of our model through numerical simulations.

1. INTRODCUTION

With the accelerating growth in data traffic, caching has been widely acknowledged as one of the most effective means to improve the performance of web applications (*e.g.*, streaming video on-demand). Storing copies of popular content at several locations in the network can greatly reduce network bandwidth usage, load on servers, and more importantly the service delay perceived by the end-users [5]. To further reduce the load on the servers, cache systems (*e.g.*, Squid [1]) usually employ techniques such as *collapsed forwarding*, where multiple user requests for the same content are aggregated and one request is sent out to the back-end server. This technique has been in use in commercial caches built at Akamai since the very early days (circa 1999) [18].

Over the past few years, the research community has been advocating for content-centric networking (CCN), an architecture that emphasizes directly accessible and routable content, arguing that today's Internet is more concerned with *what* a user wants rather than *where* it is located [15]. One important feature of the CCN architecture is the integration of caching in network routers. Named Data Networking (NDN) [12, 24] is one of the most popular designs following the CCN approach. In NDN, a user puts the name of the desired content in an *Interest* packet and sends it to the network. Once the Interest packet reaches a network node that has the content, a *Data* packet is returned to the user by traversing the reverse path taken by the Interest packet.

One of the core components of the NDN architecture is the Pending Interest Table (PIT) which performs collapsed forwarding by keeping track of currently unsatisfied Interest packets. An incoming request, at an NDN router, is forwarded to the next hop only if the PIT finds no pending Interest for the same content name. Once the content is received at the router the entry for the corresponding name is deleted from the PIT. To ensure efficient I/O operations at line speed, a fast and scalable data structure is required for the Pending Interest Table, and an accurate assessment of the PIT size is key to achieving this. Despite many experimental and numerical evaluations [10, 19, 21, 23], there has been no analytic work in modeling Pending Interest Tables. This motivates the current work.

In this paper, we analyze a cache with a Pending Interest Table, to compute the cache hit probability, response time perceived by the users, and the size of the Pending Interest Table. Unlike previous works that assume a file is instantaneously downloaded to the cache in case of a cache miss, we assume a non-zero download delay modeled as a random variable. In our analysis, we consider Time-To-Live (TTL) caches since they provide a unified framework for the analysis of single and networked caches [4, 7]. TTL caches decouple the eviction mechanisms of different files by associating each content with a timer. When a timer expires the corresponding content is evicted from the cache. TTL caches have also proven useful in accurately modeling the behavior of replacement-based caching policies such as LRU, FIFO, and RANDOM among others [8, 11, 13].

Our contributions in this paper can be summarized as follows:

- We model TTL caches with Pending Interest Tables. Our analysis is generic in the sense that it can be used to model single or networked caches assuming the request arrivals can be described as renewal processes. In our analysis, we compute the cache hit probability, mean response time,

ICN'15, September 30–October 2, 2015, San Francisco, CA, USA.
© 2015 ACM. ISBN 978-1-4503-3855-4/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810156.2810161.

and the distribution of the size of the Pending Interest Table.

- We use our model to analyze popular caching policies LRU, FIFO and RANDOM with Poisson request arrivals, and derive expressions for the cache hit probability, probability of having an entry for a content in the PIT, and the average response time. We also compute the rate at which the cache forwards requests towards the content custodian.
- We perform extensive simulations that demonstrate the accuracy of our models in predicting our metrics of interest.

The remainder of this paper is organized as follows: In the next section, we review the related work. In Section 3, we describe the system model and explain the assumptions. In Sections 4 and 5, we analyze two types of TTL caches, non-reset TTL, and reset TTL, respectively. We use the models developed in these two sections to analyze FIFO and LRU cache in Sections 6 and 7. In Section 8, we evaluate the accuracy of our model through numerical simulations. Finally, we conclude the paper in Section 9.

2. RELATED WORK

2.1 Pending Interest Table

The Pending Interest Table is one of the three fundamental data structures maintained at each router in the NDN architecture. The PIT keeps track of all the Interests that a router has forwarded but are not yet satisfied [24]. Each PIT entry stores the content name, together with the incoming and outgoing interfaces. The names are similar to URLs and are typically tens of bytes long. The URLs for pictures and videos on social networking websites, for example, require more than 80 bytes of storage [23]. A PIT is expected to store on the order of 10^7 entries on average, which makes performing operations at line speed a daunting task [22]. Over the past few years, researchers have made a great effort in design and implementation of fast and scalable Pending Interest Tables [10, 17, 19, 21, 23]. However, despite all the experimental efforts in understanding the dynamics of the PIT size [2, 20], we are unaware of any analytic work in modeling PITs. That is the problem we tackle in this paper.

2.2 Time-To-Live Caches

TTL caches, in which content eviction occurs upon the expiration of a timer, have been employed since the early days of the Internet with the Domain Name System (DNS) being an important application [13]. More recently, TTL caches have regained popularity, mostly due to admitting a general approach in the analysis of caches that can also be used to model replacement-based caching policies such as LRU. The connection between TTL caches and replacement-based (capacity-driven) policies was first established for the LRU policy by Che *et al.* [6] through the notion of cache *characteristic time*, described in Section 3.3. The characteristic time was theoretically justified and extended to other caching policies such as FIFO and RANDOM [11]. This connection was further confirmed to hold for more general arrival models than Poisson processes [16]. Over the past years, several exact and approximate analyses have been proposed for modeling single caches in isolation as well as cache networks using the TTL framework [4, 7–9]. However, there is little work that accounts for the delay in downloading files to the cache from content servers [3].

In this paper, we focus on analytic models for a single cache with a Pending Interest Table where content is fetched from the back-end server incurring some download delay. However, our model can be applied to a network of caches when request arrivals at all caches can be modeled as renewal processes.

3. MODEL DESCRIPTION

In this section, we introduce our model for a cache with requests arriving for a set of K unique files $F = \{f_1, f_2, \dots, f_K\}$ of unit size. Throughout this paper, we will use the terms content and file interchangeably. We assume that each file resides permanently at a content custodian.

Once a request arrives for a file that is in the cache, the request is served instantly. However, if the content is not found in the cache, the request will be forwarded to the content custodian. Unlike previous work that assume zero download delay, we consider the case where downloading content $f \in F$ from the custodian incurs a non-zero delay denoted by the random variable D_f. With a misuse of notation, we will use $D_f(\cdot)$ to denote the CDF of D_f. Note that since we can analyze different files individually, we can have different download delay distributions for different files.

It is assumed that the cache employs a Pending Interest Table (PIT) to aggregate requests arriving while the content is being downloaded to the cache. With the arrival of the first request to a file that is not in cache, an entry is created in the PIT. All successive requests during time D_f are aggregated at the PIT and not forwarded to the custodian. Once the content is downloaded, the PIT entry is deleted.

We consider a Time-To-Live (TTL) cache where the cache maintains a timer for each item which indicates the duration of validity for that content. The timer values could be decided by the cache or be imposed by external factors (*e.g.*, content owners). With TTL caches, the caching behaviors of different files are decoupled and thus we can consider files independently. We define T_f to denote the random variable for the TTL of content f and assume the sequence of timers are independent and identically distributed with CDF $T_f(\cdot)$.

TTL-based caching policies generally divide into two classes depending on the behavior of the TTL resets:

- *Non-reset TTL:* The timer is set once the content is downloaded to the cache, and the content leaves the cache after the timer expires.
- *Reset TTL:* The timer is set when the content is downloaded to the cache, and is reset each time a request arrives while the content is still in the cache.

Both of these policies have been properly formalized in the literature [8, 11].

3.1 Metrics of Interest

In our analysis, we are interested in computing the cache hit probability and the response time for individual files as well as the overall expected value. We will also compute the rate at which the cache forwards requests to the content custodians. Since estimating the size of the Pending Interest Table is important in the design of an appropriate data structure, we will capture the distribution of the PIT size. With TTL caches, it is also important to estimate the number of files in the cache. In order to compute the statistics of the cache and PIT size, we first compute the occupancy probability of file f in the cache and in the PIT denoted by o_f and p_f, respectively. We then define Bernoulli random variables O_f and P_f to indicate whether file f resides in cache or PIT. The random variables O_f and P_f take value one with probability o_f and p_f, and zero with probability $1-o_f$ and $1-p_f$, respectively. We then define $C=\sum_f O_f$ and $S=\sum_f P_f$ to denote the number of files in the cache and PIT respectively; they follow Poisson Binomial distributions with means $\mu_C=\sum_f o_f$ and $\mu_S=\sum_f p_f$, and variances $\sigma_C^2=\sum_f o_f(1-o_f)$ and $\sigma_S^2=\sum_f p_f(1-p_f)$, respectively. Assuming a reasonably large number of files, we can approximate the cache and PIT sizes with the Gaussian distribution with means μ_C and μ_S, and variances σ_C^2 and σ_S^2, respectively.

3.2 Renewal Arrivals

Let X_i be the time interval between the $(i-1)$st and the i-th requests for a given file. Inter-request times are assumed to be i.i.d and have distribution function $F(x)=\mathbb{P}(X_i\le x)$. Let $\lambda=1/\mathbb{E}[X_i]$ denote the arrival rate for the given file. Without loss of generality, we assume that a request for a file that is not in the cache occurs at $t=0$, *i.e.*, $X_0=0$. Let M_t denote the number of requests for the given file in the interval $(0,t]$. M_t is called the *renewal (counting) process*. Note that the request at $t=0$ is excluded, *i.e.*, $M_0=0$. Also, let

$$\tau_n=X_1+X_2+\ldots+X_n,\quad n\ge 1\quad(\tau_0=0),$$

denote the time until the arrival of the nth request. We have

$$\mathbb{P}(\tau_n\le t)=\mathbb{P}(X_1+X_2+\ldots+X_n\le t)=F^{(n)}(t),$$

where $F^{(n)}(t)$ denotes the n-fold convolution of the distribution function $F(x)$ with itself.

The expected number of renewals for the time duration $(0,t]$ is the *renewal function* $m(t)=\mathbb{E}[M_t]$, and can be expressed as

$$m(t)=\sum_{n=1}^{\infty}F^{(n)}(t),\quad t\ge 0.$$

Poisson Arrivals

A Poisson process is a renewal process with parameter λ whose inter-arrival times have the exponential distribution $F(x)=1-e^{-\lambda x}$. For a Poisson process, the renewal function simplifies to $m(t)=\lambda t$. We will use Poisson arrivals in analyzing LRU, FIFO and RANDOM caches in later sections.

3.3 Cache Characteristic Time

Che *et al.* [6] introduced the notion of *cache characteristic time*, and used it to evaluate the hit probability of a cache under the LRU policy with Poisson arrivals. Based on their work, the probability that a request for file f results in a hit can be approximated by

$$h_f=1-e^{-\lambda_f T},$$

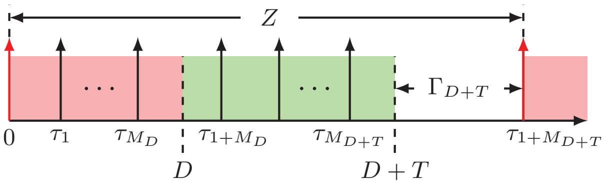


Figure 1: An entry is created in PIT for the file at time $t=0$. Content enters the cache at time $t=D$, and PIT entry is deleted. Content stays in cache until time $t=D+T$. Only requests marked as red are forwarded to the content custodian.

where λ_f is the request rate for file f, and T is a constant denoting the characteristic time of the cache and is computed as the unique solution to the equation

$$\sum_{f=1}^{K}(1-e^{-\lambda_f T})=C,$$

where C is the capacity of the LRU cache. The notion of cache characteristic time has been theoretically justified and generalized to a wider range of caching policies and more general arrival processes [16]. The idea is to compute the probability that file f occupies the cache as a function of the characteristic time $o_f(T)$, and solve for T in the fixed-point equation

$$\sum_f o_f(T)=C.$$

Cache characteristic time maps replacement-based caching policies to TTL-based caching policies. By using this notion, we will apply the analysis we develop in this paper for TTL caches to model LRU, FIFO and RANDOM caches.

4. NON-RESET TTL CACHES

In this section, we model a non-reset TTL cache with a Pending Interest Table with requests arriving according to a renewal process. In this section and the next, we model the dynamics of a single file only, as we are considering a TTL cache with no capacity constraints. We can have different distributions for download delays and TTL timers for different files, but we will use D and T without subscripts here since we are considering a specific file. In Sections 6 and 7, however, we will use subscripts to refer to the metrics of individual files, and explain how they are computed for different replacement policies subject to the cache capacity constraint.

Figure 1 shows the cache dynamics for a given file. It is assumed that initially the requested content is neither in the cache nor the PIT. The first request for the file creates an entry in PIT. It takes some time D for the file to be downloaded to the cache from a content custodian. Requests arriving for the file during D will be aggregated at the PIT, and only the first request is forwarded to the content custodian. Once the file is downloaded to the cache, the PIT entry is deleted and the TTL set to T. The content stays in the cache for T units of time until the TTL expires and the content is evicted from the cache at time $t=D+T$. This process repeats with the arrival of the next request for the file. Here, we are using generic terms D and T without subscripts for a given file, bearing in mind that different files can have different distributions for the download delay, and different values for TTL timers.

The process explained above can be divided into cycles $Z_1, Z_2, \ldots$ separated by consecutive requests sent from the cache to the custodian. These requests are marked red in Figure 1. Note that these cycles are statistically the same. Without loss of generality, consider the cycle starting at $t = 0$. The expected number of requests within this cycle equals $1 + \mathbb{E}[m(D+T)]$, where the expectation is computed with respect to the distribution of download delay D and time T. From these requests, $1 + \mathbb{E}[m(D)]$ on average result in cache misses, and will have to wait for the content to be downloaded to the cache.

Based on the above discussion, the cache hit probability of the file is

$$h = \frac{\mathbb{E}[m(D+T)] - \mathbb{E}[m(D)]}{1 + \mathbb{E}[m(D+T)]}. \tag{1}$$

Moreover, the cache and PIT occupancy probabilities for the given content can be expressed as

$$o = \frac{\mathbb{E}[T]}{\mathbb{E}[Z]} \quad \text{and} \quad p = \frac{\mathbb{E}[D]}{\mathbb{E}[Z]},$$

where $\mathbb{E}[Z]$ denotes the average cycle length. Note that o and p (defined in Section 3.1) can be used to obtain the size of the cache and PIT. To compute $\mathbb{E}[Z]$, we define Γ_t as

$$\Gamma_t := \tau_{1+M_t} - t, \tag{2}$$

to denote the time between t and the arrival of the next request. Γ_t is known as the *excess life* of the renewal process at time t, for which the complementary distribution function is expressed as (see Eq. (6.1) of Chapter 5 in [14])

$$\begin{aligned} A_\gamma(t) &= \mathbb{P}(\Gamma_t > \gamma) &(3)\\ &= 1 - F(t+\gamma) + \int_0^t \Big(1 - F(t+\gamma-x)\Big)\mathrm{d}m(x). \end{aligned}$$

We can now write the length of a cycle as

$$Z = D + T + \Gamma_{D+T},$$

and hence

$$\begin{aligned} \mathbb{E}[Z] &= \mathbb{E}[D] + \mathbb{E}[T] + \mathbb{E}[\Gamma_{D+T}] &(4)\\ &= \mathbb{E}[D] + \mathbb{E}[T] + \int_0^\infty\int_0^\infty\int_0^\infty A_x(d+t)\mathrm{d}x\mathrm{d}D(d)\mathrm{d}T(t). \end{aligned}$$

We can also compute the distribution of the cycle length as in (5). Note that $\mathbb{P}(Z \le z)$ also defines the distribution of the inter-arrival times of the requests forwarded to the content custodian, referred to as the *miss stream.* The rate of the miss stream for the file then equals $\zeta = 1/\mathbb{E}[Z]$.

As mentioned earlier, requests arriving before time D cannot be immediately served, and will experience a delayed service since the content has to be downloaded to the cache first. Let $w(t)$ denote the expected total waiting time of the requests up to time t. It is characterized by the following recursive equation

$$\begin{aligned} w(t) &= t + \int_0^t w(t-x)\mathrm{d}F(x)\\ &= t + \int_0^t (t-x)\mathrm{d}m(x), &(6) \end{aligned}$$

where the last equality follows from Theorem 4.1 of Chapter 5 in [14]. Dividing the expected total waiting time until time D, *i.e.*, $\mathbb{E}[w(D)]$, by the expected number of the requests in the cycle yields the following expression for the expected response time of the file

$$r = \frac{\mathbb{E}[w(D)]}{1 + \mathbb{E}[m(D+T)]}.$$

5. RESET TTL CACHES

In this section, we analyze a TTL cache where the TTL is reset every time a request arrives for a file that is in the cache. Figure 2 illustrates the cache dynamics with requests arriving for a given content.

With the arrival of the first missed request an entry is created for the file in PIT, and consecutive requests for the file are aggregated until content is downloaded to the cache at time D. When the content enters the cache, the TTL is set to T_0. If the next request arrives after the TTL expires, *i.e.*, $\Gamma_D > T_0$, the content will be evicted from cache at time $t = D + T_0$, similar to the non-reset TTL case. However, if a request arrives before the TTL expires, the TTL will be reset to time T_1. For notational simplicity, we define Y_n to denote the inter-arrival time for the request after the nth hit. Note that Y_n follows the same distribution as X_i and has CDF $F(\cdot)$. The file will remain in the cache as long as successive requests arrive no later than the TTL expires, *i.e.*, $Y_n \le T_n$. We assume that the sequence of the TTL timers $T_0, T_1, \ldots$ are independent and identically distributed according to $T(\cdot)$, and independent of the request inter-arrival times, Y_n.

For Γ_D we can use (3) to get

$$\begin{aligned} \mathbb{P}(\Gamma_D \le \gamma \mid D = d) &= 1 - A_\gamma(d) &(7)\\ &= F(d+\gamma) - \int_0^d \Big(1 - F(d+\gamma-x)\Big)\mathrm{d}m(x). \end{aligned}$$

Note that if $\Gamma_D > T_0$, the cycle Z ends at τ_{1+M_D} with no cache hits. With $\Gamma_D \le T_0$ we observe the first cache hit. The second hit occurs if $Y_1 \le T_1$. Assuming that exactly N requests result in cache hits, we must have $Y_n \le T_n$, $n = 1, 2, \ldots, N-1$, and $Y_N > T_N$. The distribution of the number of cache hits, hence, resembles a geometric distribution. We have

$$\mathbb{P}(N = n \mid D = d) = \begin{cases} q_\Gamma & n = 0,\\ (1-q_\Gamma)(1-q_T)^{n-1}q_T & n \ge 1, \end{cases}$$

where

$$q_\Gamma = 1 - \int_0^\infty \mathbb{P}(\Gamma_d \le t)\mathrm{d}T(t),$$

denotes the probability of no cache hits, and

$$q_T = 1 - \int_0^\infty F(t)\mathrm{d}T(t),$$

is the probability $\mathbb{P}(Y_i > T_i), i = 1, 2, \ldots, n-1$. This yields

$$\mathbb{E}[N \mid D = d] = (1 - q_\Gamma)/q_T.$$

The expected number of cache hits then equals

$$\mathbb{E}[N] = \frac{\int_0^\infty\int_0^\infty \mathbb{P}(\Gamma_d \le t)\mathrm{d}T(t)\mathrm{d}D(d)}{1 - \int_0^\infty F(t)\mathrm{d}T(t)}. \tag{8}$$

We can now write the cache hit probability for the given content as the expected number of hit requests divided by the expected total number of requests,

$$h = \frac{\mathbb{E}[N]}{1 + \mathbb{E}[m(D)] + \mathbb{E}[N]}. \tag{9}$$

Also, assuming N cache hits occur in a cycle, the cycle duration Z can be expressed as

$$Z = \sum_{i=1}^{1+M_D} X_i + \sum_{n=1}^{N} Y_n.$$

Note that N is a stopping time with respect to the sequence $\{T_n - Y_n\}$, corresponding to the rule "stop as soon as $T_n - Y_n \leq 0$." By applying Wald's equation we get

$$\mathbb{E}\left[\sum_{n=1}^{N} Y_n\right] = \mathbb{E}[Y_1]\mathbb{E}[N] = \mathbb{E}[X]\mathbb{E}[N] = \mathbb{E}[N]/\lambda,$$

and therefore

$$\mathbb{E}[Z] = \Big(1 + \mathbb{E}[m(D)] + \mathbb{E}[N]\Big)/\lambda. \tag{10}$$

The rate of the miss stream then is

$$\zeta = \frac{1}{\mathbb{E}[Z]} = \frac{\lambda}{1 + \mathbb{E}[m(D)] + \mathbb{E}[N]}.$$

We can also write the cache occupancy probability for the content as the expected time spent in cache, colored green in Figure 2, divided by the expected cycle length as

$$o = \frac{\mathbb{E}[\Gamma_D] + \mathbb{E}[N]/\lambda - \mathbb{E}[\Gamma_{t_E}]}{\mathbb{E}[Z]},$$

where t_E denotes the eviction time of the content. It is easy to see, by comparing the above equation with (9), that $h = o$ if the request arrival process is Poisson, since with exponential inter-arrival times we have $\mathbb{E}[\Gamma_D] = \mathbb{E}[\Gamma_{t_E}] = \mathbb{E}[\Gamma_t], \forall t$.

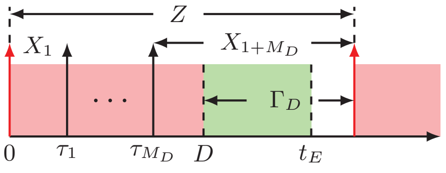
We can also write the probability of having an entry in the PIT for the file as

$$p = \frac{\mathbb{E}[D]}{\mathbb{E}[Z]} = \frac{\lambda\mathbb{E}[D]}{1 + \mathbb{E}[m(D)] + \mathbb{E}[N]}. \tag{11}$$

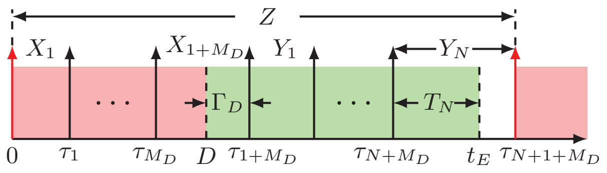
As explained in Section 3.1, we can use o and p to compute statistics regarding the number of items in the cache and PIT.

Computing the distribution of the cycle length requires some care. Note that making an assumption on the number of hits affects the distribution of Γ_D. Specifically, let $H_0^d(\cdot)$ denote the distribution of Γ_d (assuming $D = d$) when we do not get any hits in a cycle. We have

$$\begin{aligned} H_0^d(\gamma) &= \int_0^\infty \mathbb{P}(\Gamma_d \leq \gamma \mid \Gamma_d > T_0, T_0 = t)\mathrm{d}T(t) \\ &= \Big(1 - A_\gamma(d)\Big)\int_0^\gamma \frac{\mathrm{d}T(t)}{A_t(d)}. \end{aligned}$$



(a) $\Gamma_D > T_0$



(b) $\Gamma_D \leq T_0$

Figure 2: An entry is created in PIT for the file at time $t = 0$. Content enters the cache at time $t = D$, and PIT entry is deleted. (a) The content will be evicted from cache at time $t_E = D + T_0$ if $\Gamma_D > T_0$. (b) The TTL will be reset if $\Gamma_D \leq T_0$, and the content will continue to stay in the cache as long as $Y_n \leq T_n$. Here, $t_E = \tau_{N+M_D} + T_N$ denotes the time that the content is evicted from cache.

Similarly, let $H_1^d(\cdot)$ denote the distribution of Γ_d when there is at least one hit in a cycle. We have

$$\begin{aligned} H_1^d(\gamma) &= \int_0^\infty \mathbb{P}(\Gamma_d \leq \gamma \mid \Gamma_d \leq T_0, T_0 = t)\mathrm{d}T(t) \\ &= \Big(1 - A_\gamma(d)\Big)\int_\gamma^\infty \frac{\mathrm{d}T(t)}{1 - A_t(d)}. \end{aligned}$$

Moreover, assuming we get N hits in a cycle translates into $Y_1, Y_2, \ldots, Y_{N-1}$ having the distribution

$$\begin{aligned} L_1(y) &= \int_0^\infty \mathbb{P}(Y_n \leq y \mid Y_n \leq T_n, T_n = t)\mathrm{d}T(t) \\ &= F(y)\int_y^\infty \frac{\mathrm{d}T(t)}{F(t)}, \end{aligned}$$

and for Y_N we have

$$\begin{aligned} L_2(y) &= \int_0^\infty \mathbb{P}(Y_N \leq y \mid Y_N > T_N, T_N = t)\mathrm{d}T(t) \\ &= F(y)\int_0^y \frac{\mathrm{d}T(t)}{1 - F(t)}. \end{aligned}$$

The cycle length can be expressed as

$$Z = D + \Gamma_D + \sum_{n=1}^{N} Y_n,$$

and since Γ_D and $Y_n, n = 1, \ldots, N$ are conditionally independent given N and D, we can write the distribution

---

$$\begin{aligned} \mathbb{P}(Z \leq z) &= \int_0^z \int_0^{z-t} \mathbb{P}(D + T + \Gamma_{D+T} \leq z \mid D = d, T = t)\mathrm{d}D(d)\mathrm{d}T(t) \\ &= \int_0^z \int_0^{z-t} \Big(1 - A_{z-d-t}(d + t)\Big)\mathrm{d}D(d)\mathrm{d}T(t). \end{aligned} \tag{5}$$

function of Z as

$$\mathbb{P}(Z \leq z) = \int_0^z \Big[H_0^d(z)\mathbb{P}(N=0 \mid D=d) \qquad (12)$$
$$+ \sum_{n=1}^{\infty} \Big(H_1^d * L_2 * L_1^{(n-1)}\Big)(z)\mathbb{P}(N=n \mid D=d)\Big]\mathrm{d}D(d),$$

where $(f*g)(\cdot)$ denotes the convolution of f and g, and $L_1^{(n)}$ denotes the n-fold convolution of L_1 with itself.

The service delay experienced by requests arriving before time D will be the same as the non-reset TTL case, and we have

$$w(t) = t + \int_0^t (t-x)\mathrm{d}m(x).$$

The expected response time of the file then equals

$$r = \frac{\mathbb{E}[w(D)]}{1+\mathbb{E}[m(D)]+\mathbb{E}[N]}. \qquad (13)$$

6. FIFO WITH POISSON ARRIVALS

In this section, we consider a FIFO cache of size C with requests arriving to the cache according to a Poisson process. We assume that requests for file f arrive with rate λ_f. It was shown in [7] that a FIFO cache can be modeled as a TTL cache with constant non-reset timers $T_f = T, \forall f$. The constant T is the characteristic time of the FIFO cache; we will explain how to compute T in the remainder of the section.

6.1 Cache Hit Probability

With a Poisson process, the renewal function for file f is $m_f(t) = \lambda_f t$, and hence using (1) we can write the cache hit probability as

$$h_f = \frac{\mathbb{E}[\lambda_f(D_f+T)] - \mathbb{E}[\lambda_f D_f]}{1+\mathbb{E}[\lambda_f(D_f+T)]}$$
$$= \frac{\lambda_f T}{1+\lambda_f(\mathbb{E}[D_f]+T)}. \qquad (14)$$

Note that when $\mathbb{E}[D_f]=0$, we obtain

$$h_f = \lambda_f T/(1+\lambda_f T),$$

which is the expression obtained in [7] for the hit probability of a FIFO cache under the zero download delay assumption. With Poisson arrivals, the cache occupancy probability equals the cache hit probability, *i.e.*, $o_f = h_f$, and hence as explained in Section 3.3, the value of T can be computed by solving the fixed-point equation

$$\sum_{f=1}^{K} \frac{\lambda_f T}{1+\lambda_f(\mathbb{E}[D_f]+T)} = C.$$

6.2 Size of Pending Interest Table

The expected cycle length can be computed from (4), and for Poisson processes is

$$\mathbb{E}[Z_f] = \mathbb{E}[D_f] + T + 1/\lambda_f.$$

The probability of having an entry for file f in the PIT then is

$$p_f = \frac{\lambda_f \mathbb{E}[D_f]}{1+\lambda_f(\mathbb{E}[D_f]+T)}. \qquad (15)$$

As explained in Section 3.1, the size of the Pending Interest Table, S, can be approximated as a Gaussian random variable with mean $\mu_S = \sum_f p_f$ and variance $\sigma_S^2 = \sum_f p_f(1-p_f)$.

6.3 Cache Response Time

Using $m_f(t) = \lambda_f t$ for Poisson processes in (6), the total waiting time of requests until time t can be written as $w_f(t) = t + 0.5\lambda_f t^2$. Therefore, the expected response time for file f equals

$$r_f = \frac{\mathbb{E}[D_f + 0.5\lambda_f D_f^2]}{1+\lambda_f(\mathbb{E}[D_f]+T)}.$$

For the cases of deterministic and exponentially distributed download delays, we can simplify the response time as follows:

- If the delay to download a content to the cache is deterministic, the average response time equals

$$r_f = \frac{D_f + 0.5\lambda_f D_f^2}{1+\lambda_f(D_f+T)}.$$

- If the download delay follows an exponential distribution, for the expected response time of file f we obtain

$$r_f = \frac{\mathbb{E}[D_f] + \lambda_f \mathbb{E}^2[D_f]}{1+\lambda_f(\mathbb{E}[D_f]+T)}.$$

6.4 Miss Process

The expected cycle length can be computed from (4) and equals

$$\mathbb{E}[Z_f] = \Big(1+\lambda_f(\mathbb{E}[D_f]+T)\Big)/\lambda_f.$$

The rate at which the cache forwards requests for file f to the custodian then is

$$\zeta_f = 1/\mathbb{E}[Z_f] = \lambda_f/\Big(1+\lambda_f(\mathbb{E}[D_f]+T)\Big).$$

We can also compute the distribution of the cycle lengths. First note that for Poisson processes we have

$$A_\gamma(t) = e^{-\lambda_f \gamma}.$$

Therefore, from (5) we obtain

$$\mathbb{P}(Z_f \leq z) = \int_0^{z-T} \Big(1 - e^{-\lambda_f(z-d-T)}\Big)\,\mathrm{d}D_f(d),$$

which for deterministic and exponentially distributed download delays can be simplified as follows:

- For deterministic download delays

$$\mathbb{P}(Z_f \leq z) = \begin{cases} 0, & z < D_f + T, \\ 1 - e^{-\lambda_f(z-D_f-T)}, & z \geq D_f + T. \end{cases}$$

- For exponentially distributed download delays

$$\mathbb{P}(Z_f \leq z) = \begin{cases} 0, & z < T, \\ 1 - e^{-(z-T)/\mathbb{E}[D_f]} \\ \quad - \dfrac{e^{-(z-T)/\mathbb{E}[D_f]} - e^{-\lambda_f(z-T)}}{\lambda_f \mathbb{E}[D_f] - 1}, & z \geq T. \end{cases}$$

6.5 RANDOM with Poisson Arrivals

It was shown in [7] that a RANDOM cache can be modeled as a TTL cache with exponentially distributed non-reset timers. Repeating the analysis we did in this section for exponentially distributed T with mean $\mathbb{E}[T]$ reveals that all the expressions derived for a FIFO cache can be used for a RANDOM cache by replacing T with $\mathbb{E}[T]$. The only expression that is different is the distribution function for the cycle length $\mathbb{P}(Z_f \leq z)$. For deterministic and exponentially distributed download delays we have

- For deterministic download delays

$$\mathbb{P}(Z_f \leq z) = \begin{cases} 0, & z < D_f, \\ 1 - e^{-(z-D_f)/\mathbb{E}[T]} \\ \quad - \dfrac{e^{-(z-D_f)/\mathbb{E}[T]} - e^{-\lambda_f (z-D_f)}}{\lambda_f \mathbb{E}[T] - 1}, & z \geq D_f. \end{cases}$$

- For exponentially distributed download delays, note that D_f, T and Γ_{D_f+T} follow independent exponential distributions. Note that with Poisson arrivals, Γ_{D_f+T} has the same distribution as the inter-arrival times for file f, and hence has CDF $F_f(\cdot)$. Therefore, the distribution of the cycle length follows as

$$\mathbb{P}(Z_f \leq z) = (D_f * T * F_f)(z), \; z \geq 0.$$

7. LRU WITH POISSON ARRIVALS

In this section, we model an LRU cache of capacity C with requests arriving to the cache according to a Poisson process. We assume that requests for file f arrive with rate λ_f. It was shown in [7] that an LRU cache can be modeled as a TTL reset cache with constant timers $T_f = T, \forall f$.

7.1 Cache Hit Probability

First, we note that with Poisson arrivals we have

$$\mathbb{P}(\Gamma_d \leq \gamma) = 1 - e^{-\lambda_f \gamma},$$

which is independent of d. The above equation suggests that the expected number of cache hits in a cycle is independent of the distribution of the download delay. Using (8), we obtain

$$\mathbb{E}[N_f] = \frac{1 - e^{-\lambda_f T}}{e^{-\lambda_f T}} = e^{\lambda_f T} - 1.$$

Based on (9), the hit probability of content f can then be written as

$$h_f = \frac{e^{\lambda_f T} - 1}{\lambda_f \mathbb{E}[D] + e^{\lambda_f T}}.$$

Note that $\mathbb{E}[D_f] = 0$ yields $h_f = 1 - e^{-\lambda_f T}$ which is the expression obtained by Che *et al.* [6] for the hit probability of an LRU cache under the assumption of zero download delay.

For Poisson arrivals the cache occupancy probability equals the cache hit probability, *i.e.*, $o_f = h_f$, and hence the value of T is obtained by solving the fixed-point equation

$$\sum_{f=1}^{K} \frac{e^{\lambda_f T} - 1}{\lambda_f \mathbb{E}[D_f] + e^{\lambda_f T}} = C.$$

7.2 Size of Pending Interest Table

Based on (10), the expected cycle length equals

$$\mathbb{E}[Z_f] = \mathbb{E}[D_f] + e^{\lambda_f T}/\lambda_f,$$

and hence using (11), the probability of having an entry for file f in the PIT is given by

$$p_f = \frac{\lambda_f \mathbb{E}[D_f]}{\lambda_f \mathbb{E}[D_f] + e^{\lambda_f T}}.$$

As explained in Section 3.1, the size of the Pending Interest Table, S, can be approximated as a Gaussian with mean $\mu_S = \sum_f p_f$ and variance $\sigma_S^2 = \sum_f p_f(1 - p_f)$.

7.3 Cache Response Time

Considering Poisson processes, the total waiting time of requests until time t (for $t \leq D_f$) is $w_f(t) = t + 0.5\lambda_f t^2$, and using (13) the average response time for file f is

$$r_f = \frac{\mathbb{E}[D_f + 0.5\lambda_f D_f^2]}{\lambda_f \mathbb{E}[D_f] + e^{\lambda_f T}}.$$

For deterministic and exponentially distributed download delays, we can simplify the response time as follows:

- For deterministic download delays, we can simply write the expected response time for file f as

$$r_f = \frac{D_f + 0.5\lambda_f D_f^2}{\lambda_f D_f + e^{\lambda_f T}}.$$

- If the download delay is exponentially distributed, the expected delay simplifies to

$$r_f = \frac{\mathbb{E}[D_f] + \lambda_f \mathbb{E}^2[D_f]}{\lambda_f \mathbb{E}[D_f] + e^{\lambda_f T}}.$$

7.4 Miss Process

The rate at which requests get forwarded to the custodian equals

$$\zeta_f = \frac{1}{\mathbb{E}[Z_f]} = \frac{\lambda_f}{\lambda_f \mathbb{E}[D_f] + e^{\lambda_f T}}.$$

8. PERFORMANCE EVALUATION

In this section, we evaluate the accuracy of our models by comparing against numerical simulations. We simulate LRU and FIFO caches of size $C = 1000$, where requests arrive for $N = 10,000$ files. File popularities follow a Zipf distribution with parameter $\alpha = 0.8$, *i.e.*, $\lambda_f \propto 1/f^\alpha$, and the aggregate request rate is assumed to be $\lambda = 10^5$ requests per second. In evaluating the accuracy of the models for individual files, the download delay is set to 100ms. We also compute the average values for different metrics by varying the download delay from zero to 250ms.

8.1 Cache Hit Probability

Figure 3 shows the hit probability of the individual files for LRU and FIFO caches. The hit probability computed by the original characteristic time approximation assuming zero download delay (ZDD) [6] is also shown for comparison. While our models accurately estimate the hit probabilities of the individual files for the LRU and FIFO caches, it is clear that with ZDD assumption the behavior of the system cannot be captured.

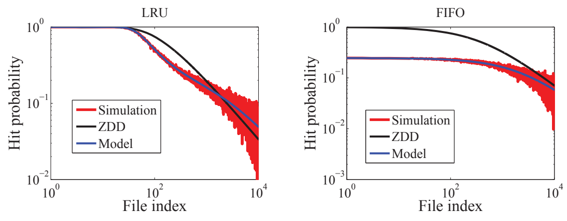


Figure 3: Cache hit probability for individual files.

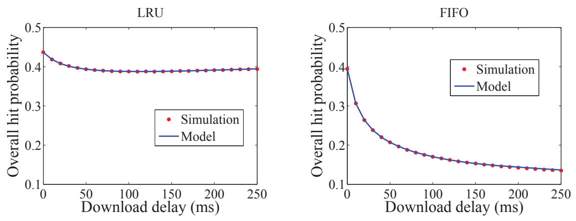


Figure 4: Overall cache hit probability.

Figure 4 shows the overall cache hit probability computed, from simulations and our models, for different values of the download delay. This figure suggests that increasing the download delay drastically affects the hit probability of the FIFO cache, while it has minor impact on the hit probability of the LRU cache.

8.2 Size of Pending Interest Table

Figure 5 shows the probability of having an entry for each file in the Pending Interest Table. Note that for Poisson arrivals this also equals the probability that a request will be aggregated in the PIT and not forwarded to the server.

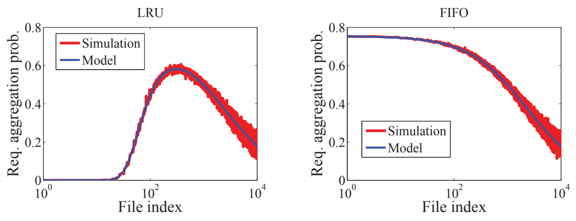


Figure 5: Request aggregation probability in PIT.

In Section 3.1, we advocated the use of the Gaussian distribution to approximate the size of the Pending Interest Table. Figure 6 shows the distribution of the PIT size computed from simulations, as well as a Gaussian distributions with moments computed based on the discussion in Section 3.1. It is clear that the Gaussian distribution accurately represents the distribution of the PIT size for both LRU and FIFO caches. Figure 6 also suggests that a FIFO cache yields a larger Pending Interest Table compared to an LRU cache of the same size.

Figure 7 shows the average PIT size for different values of the cache download delay. As one might expect, as download delay increases, the size of the Pending Interest Table increases, and our model accurately predicts the expected PIT size.

8.3 Cache Response Time

Figure 8 shows the average response time for individual files when the cache download delay is $D = 100$ms. With the LRU policy, the average response times for the most popular files are zero, which suggests that these files are almost always in the cache. With the FIFO cache, however, even the most popular file has a non-zero average response time. This is due to the non-reset TTL nature of the FIFO policy, *i.e.*, even the most popular file gets evicted from cache T_{FIFO} time after the insertion into the cache, where T_{FIFO} denotes the characteristic time of the FIFO cache.

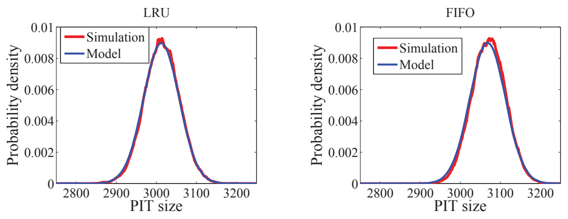


Figure 6: PIT size distribution.

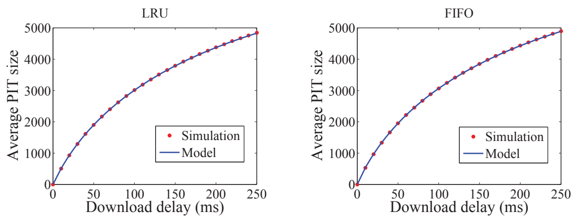


Figure 7: Average PIT size.

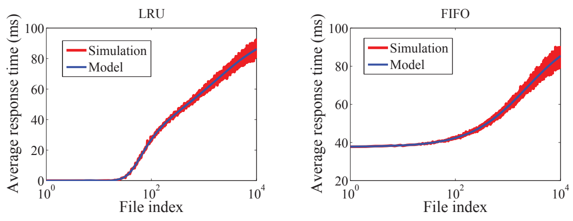


Figure 8: Response time per file.

Figure 9 shows how average response time depends on download delay. Comparing the two plots in Figure 9 we conclude that the LRU cache yields lower average response times compared to the FIFO cache. In fact, for large download delay, LRU achieves a 30% lower average response time.

8.4 Miss Request Forwarding

Figure 10 shows the probability of forwarding a request to other network nodes for each file, and Figure 11 shows the rate at which requests are forwarded. While with an LRU cache, requests for the most popular file are almost never forwarded, with a FIFO cache, the most popular file has the highest request forwarding rate. Figure 11 looks like Figure 5 because the expressions for PIT hit probability and request forwarding rate are related through $p_f = \zeta_f \mathbb{E}[D_f]$.

Figure 12 shows the overall request forwarding rate as a function of download delay. As download delay increases, more requests are aggregated at the PIT and hence fewer requests are forwarded.

8.5 A Larger System

Convinced of the accuracy of our models, we use them to analyze properties of a larger system. We consider a system with $N = 10^7$ files, where file popularities follow

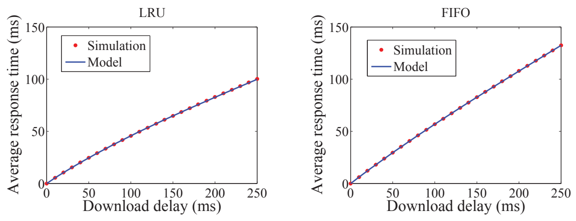


Figure 9: Average response time.

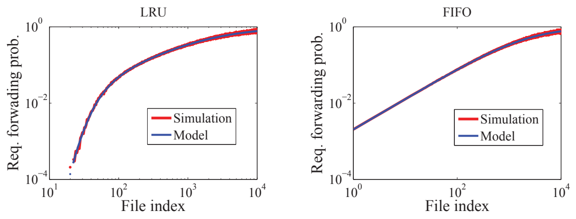


Figure 10: Request forwarding probability for individual files.

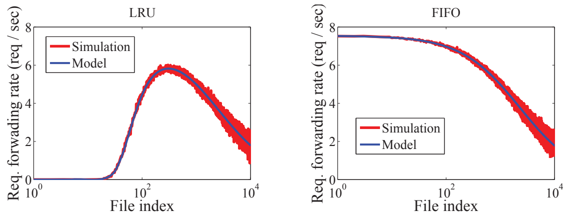


Figure 11: Request forwarding rate for individual files.

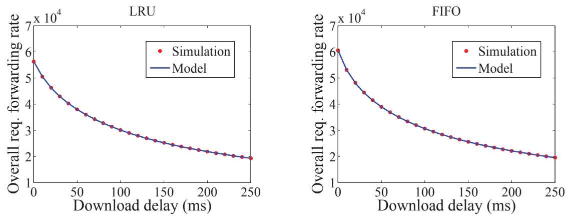


Figure 12: Overall request forwarding rate.

a Zipf distribution with parameter $\alpha = 0.8$. The aggregate request rate at the cache is $\lambda = 10^5$ requests per second. We compute the average cache hit probability, average response time, average PIT size, and the average request forwarding rate for various cache sizes and download delays. When studying the effect of the cache size we set the download delay to $D = 100$ms, and when exploring the effect of the download delay we set the cache size equal to $C = 10^4$. Here, we only present results from the models as it takes a significantly long time to simulate the system explained above.

Figure 13 shows the average hit probability for the LRU and FIFO caches for various values of cache size and download delay. Although the plots seem to be very close for LRU and FIFO caches, the LRU cache achieves up to 60% higher hit probability for some cache sizes. Moreover, this difference increases as download delay increases.

Figure 14 demonstrates the effect of the cache size and download delay on the average response time of the LRU and FIFO caches. For small and very large cache sizes LRU and FIFO caches exhibit similar performance. However, for mid-sized caches, LRU yields up to 20% lower response time. For a given cache size, the response times of the LRU and FIFO caches show linear increase as download delay increases, and the two caches exhibit similar performance.

Figure 15 shows how the average PIT size changes with cache size and download delay. For small and very large cache sizes, the average PIT size is almost the same for LRU and FIFO caches, while for mid-sized caches, the LRU cache saves up to 20% on the PIT size. With a given cache size, the PIT size shows a linear increase with the download delay, and the difference in the PIT sizes of the LRU and FIFO caches is less than 1% for the values of the download delays explored here.

Figure 16 shows the effect of cache size and download delay on the overall request forwarding rate. The average request forwarding rate behaves very similar to response time with respect to the cache size. For small and very large cache sizes the LRU and FIFO policies yield almost the same forwarding rates but for some mid-sized caches up to 18% reduction in forwarding rate can be achieved by using an LRU cache. As one might expect, the request forwarding rate decreases, as the download delay is increased.

9. CONCLUSION

In this paper, we consider the problem of modeling a cache with a Pending Interest Table. It is assumed that in case of a cache miss, it would take some time to download the content to the cache, where the download delay is modeled as a random variable. While the content is being downloaded to the cache, requests arriving for the content are aggregate at the Pending Interest Table, and not forwarded to the content custodian in order to reduce the load on the server. We derive expressions for the cache hit probability, response time perceived by the users, and the size of the Pending Interest Table. We analyze two classes of TTL-based caching policies, where the timer could be set only once, or be reset with every request for the content. Our analysis enables us to model the traditional replacement-based caching policies LRU, FIFO and RANDOM. We perform numerical simulations that demonstrate the accuracy of our approach in modeling caches with Pending Interest Tables.

Acknowledgements

This work was conducted under an NSF grant CNS-1413998 and a MURI ARO grant W911NF-12-10385. Ali Dabirmoghaddam was supported by a UCSC Chancellor's Dissertation-Year Fellowship.

10. REFERENCES

[1] Squid. `squid-cache.org`.

[2] Abu, A. J., Bensaou, B., and Wang, J. M. Interest packets retransmission in lossy ccn networks and its impact on network performance. In *ICN* (2014), pp. 167–176.

[3] Bahat, O., and Makowski, A. M. Measuring consistency in ttl-based caches. *Performance Evaluation 62* (2005), 439–455.

[4] Berger, D. S., Gland, P., Singla, S., and Ciucu, F. Exact analysis of ttl cache networks. *Performance Evaluation 79* (2014), 2–23.

[5] Borst, S., Gupta, V., and Walid, A. Distributed caching algorithms for content distribution networks. In *INFOCOM* (March 2010), pp. 1–9.

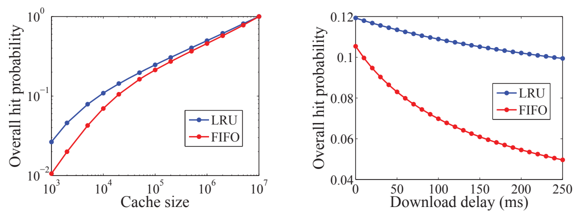


Figure 13: Overall hit probability.

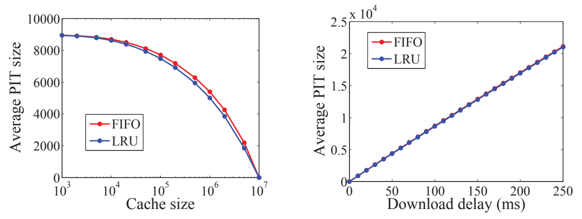


Figure 15: Average PIT size.

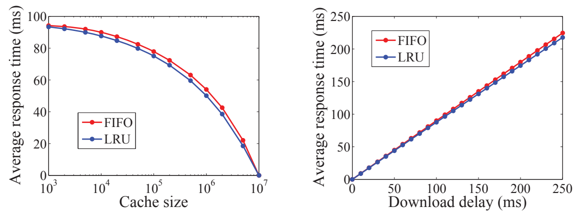


Figure 14: Average response time.

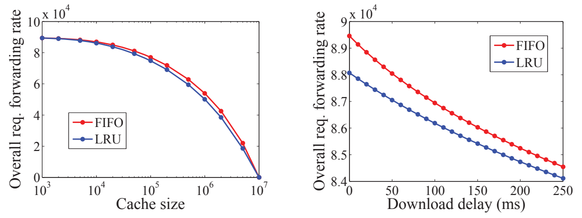


Figure 16: Overall request forwarding rate.

[6] Che, H., Wang, Z., and Tung, Y. Analysis and design of hierarchical web caching systems. In *INFOCOM* (2001), pp. 1416–1424.

[7] Choungmo-Fofack, N., Dehghan, M., Towsley, D., Badov, M., and Goeckel, D. L. On the performance of general cache networks. In *ValueTools* (December 2014).

[8] Choungmo-Fofack, N., Nain, P., Neglia, G., and Towsley, D. Analysis of ttl-based cache networks. In *ValueTools* (October 2012), pp. 1–10.

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

[10] Dai, H., Liu, B., Chen, Y., and Wang, Y. On pending interest table in named data networking. In *ANCS* (2012), pp. 211–222.

[11] Fricker, C., Robert, P., and Roberts, J. A versatile and accurate approximation for lru cache performance. In *ITC* (2012).

[12] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. Networking named content. In *CoNEXT* (2009), pp. 1–12.

[13] Jaeyeon, J., Berger, A. W., and Balakrishnan, H. Modeling ttl-based internet caches. In *INFOCOM* (March 2003), pp. 417–426.

[14] Karlin, and Samuel. *A first course in stochastic processes*, 2 ed. Academic press, 1975.

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

[16] Martina, V., Garetto, M., and Leonardi, E. A unified approach to the performance analysis of caching systems. In *INFOCOM* (April 2014), pp. 2040–2048.

[17] Perino, D., and Varvello, M. A reality check for content centric networking. In *SIGCOMM Workshop on Information-centric Networking* (2011), pp. 44–49.

[18] Sitaraman, R. personal communication, April, 2015.

[19] Varvello, M., Perino, D., and Linguaglossa, L. On the design and implementation of a wire-speed pending interest table. In *INFOCOM workshops* (April 2013), pp. 369–374.

[20] Virgilio, M., Marchetto, G., and Sisto, R. Pit overload analysis in content centric networks. In *SIGCOMM Workshop on Information-centric Networking* (2013), pp. 67–72.

[21] Wang, Y., He, K., Dai, H., Meng, W., Jiang, J., Liu, B., and Chen, Y. Scalable name lookup in ndn using effective name component encoding. In *ICDCS* (2012), pp. 688–697.

[22] You, W., Mathieu, B., Truong, P., Peltier, J., and Simon, G. Dipit: A distributed bloom-filter based pit table for ccn nodes. In *ICCCN* (July 2012), pp. 1–7.

[23] Yuan, H., and Crowley, P. Scalable pending interest table design: From principles to practice. In *INFOCOM* (April 2014), pp. 2049–2057.

[24] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. Named data networking. *ACM SIGCOMM Computer Communication Review 44* (July 2014), 66–73.