Estimating DNS Source and Cache Dynamics under Interval-Censored Age Sampling

DI XIAO, Texas A&M University, College Station, United States
XIAOYONG LI, Texas A&M University, College Station, United States
DAREN CLINE, Texas A&M University, College Station, United States
DMITRI LOGUINOV, Texas A&M University, College Station, United States

Since inception, DNS has used a TTL-based replication scheme that allows the source (i.e., an authoritative domain server) to control the frequency of record eviction from client caches. Existing studies of DNS predominantly focus on reducing query latency and source bandwidth, both of which are optimized by increasing the cache hit rate. However, this causes less-frequent contacts with the source and results in higher staleness of retrieved records. Given high data-churn rates at certain providers (e.g., dynamic DNS, CDNs) and importance of consistency to their clients, we propose that cache models include the probability of freshness as an integral performance measure. We derive this metric under general update/download processes and present a novel framework for measuring its value using remote observation (i.e., without access to the source or the cache). Besides freshness, our methods can estimate the inter-update distribution of DNS records, cache hit rate, distribution of TTL, and query arrival rate from other clients. Furthermore, these algorithms do not require any changes to the existing infrastructure/protocols.

CCS Concepts: • **Networks** → **Network performance modeling;**

Additional Key Words and Phrases: Cache staleness, TTL sampling, lifetime estimation

ACM Reference Format:
Di Xiao, Xiaoyong Li, Daren Cline, and Dmitri Loguinov. 2025. Estimating DNS Source and Cache Dynamics under Interval-Censored Age Sampling. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 10, 3, Article 13 (May 2025), 29 pages. https://doi.org/10.1145/3712697

1 Introduction

To keep up with the explosive growth of Internet traffic, end-to-end caches continue to be an important part of many distributed systems, including search engines [3, 8, 11, 44, 47], wireless mobile networks [23, 24], P2P structures [46, 50], **Content Distribution Networks (CDNs)** [7, 36], **Domain Name System (DNS)** [10, 31, 41], data warehouses [49], and various web applications [18, 20, 33, 48]. If **Information-Centric Networking (ICN)** [1] becomes successful, the Internet may eventually see cache deployment even at the network layer (i.e., at each router). Therefore,

An earlier version of the article appeared in IEEE INFOCOM 2018.
Authors' Contact Information: Di Xiao, Texas A&M University, College Station, Texas, United States; e-mail: di@cse.tamu.edu; Xiaoyong Li, Texas A&M University, College Station, Texas, United States; e-mail: xiaoyong@cse.tamu.edu; Daren Cline, Texas A&M University, College Station, Texas, United States; e-mail: dcline@stat.tamu.edu; Dmitri Loguinov, Texas A&M University, College Station, Texas, United States; e-mail: dmitri@cs.tamu.edu.

ACM 2376-3639/2025/05-ART13
https://doi.org/10.1145/3712697

modeling cache performance is crucial to our current and future understanding of data churn at origin servers, Internet core, and customer facilities, including such metrics as bandwidth consumption, data consistency (i.e., freshness), and latency.

Depending on the eviction policy, cache operation can be classified into *capacity-based* and **time to live** ***(TTL)****-based*. In the former case, arrival of new items into a full cache causes immediate removal of elements deemed unpopular (e.g., using LRU, LFU, CLOCK, FIFO, Random, and their variations [38]). In the latter case, which is our focus in this article, items are evicted only when their TTL expires, meaning that cache size is considered infinite. This approach is more suitable in scenarios where object *staleness*, rather than storage limitations, is of primary concern and the duration an item remains cached must be controlled by the source based on the record's churn rate (i.e., frequency of modification).

One particular area, where TTL-based caching has long been part of the standard, is the DNS. With the wide adoption of dynamic DNS services and proliferation of CDN, many authoritative domains now frequently change IP addresses and other records to reflect the current server location, content availability, traffic load, and routing/geographic preferences, with more such activity expected in the future [16]. While the cache hit ratio h has been the sole metric of performance for many years [5, 6, 19, 25, 26, 28, 29, 31], the modern Internet requires a different modeling objective that would balance record *freshness* against cache *efficiency*. In this context, simply maximizing the hit rate, which essentially means setting the TTL to infinity, is not a meaningful pursuit. Instead, the system involves a tradeoff – higher hit rates h require items to stay longer in the cache, while better freshness f entails the opposite. Unfortunately, the interplay between these metrics has not received much attention in the past.

In order to keep staleness below target levels, as well as achieve accurate performance characterization, one requires a methodology for estimating the various parameters of the system (such as f) within the confines of the current DNS protocol. This process, which we call *remote measurement*, must obtain the various hidden distributions without having direct access to ground truth. Assuming $U_1, U_2, \ldots,$ are random inter-update delays of a given record, the main question considered in this work is how to estimate their distribution $F_U(x)$ without access to the source. We consider two measurement scenarios— *passive*, which monitors ongoing traffic at the cache without generating new queries, and *active*, which does not have access to the cache, but is allowed to send non-intrusive (i.e., iterative) probes that ask for the already-cached content (i.e., recursive queries that cause cache pollution are not allowed). In the passive case, we show that the sampling process at time t_k can bound the delay since the last update, i.e., age $A_U(t_k)$, to some deterministic interval $[L_k, R_k]$. This type of data truncation is known as *interval censoring* [51], where previous work is mostly limited to observations of the target variable itself (i.e., inter-update delay U_i) rather than its age $A_U(t_k)$. As a consequence, these approaches are not concerned with the fact that a valid age distribution $G_U(x)$ must be *concave*, which precludes results of these methods from being meaningful in convolutions involving $G_U(x)$ or attempts to reconstruct $F_U(x)$. Instead, this problem requires new algorithms for building inherently concave distributions out of interval-censored age samples. For the active case, the main difference is that both L_k, R_k are skewed by certain random parameters of the cache (e.g., TTLs) whose distribution needs to be estimated separately and the bias corrected using additional probabilistic techniques (i.e., inference of process age from its observed residuals).

All of this makes sampling f a formidable challenge. While existing studies [2, 15, 37, 42, 52] provide a framework for estimating h and the average user-request rate λ, these methods cannot be easily extended to handle freshness and their models do not accommodate the possibility of random TTLs. Both are important factors that we aim to address below.

1.1 Main Results

Our first contribution is to create a model of system operation using general point processes and define a new metric of performance – probability f that an incoming query to the local resolver encounters a fresh (i.e., consistent with the source) copy of the record. We model updates at the authoritative DNS server as some point process N_U, client requests to the local resolver as a renewal process N_Q, and TTL values $\{T_k\}$ as random variables with some distribution $F_T(x)$. We not only derive f under these conditions, but also correct an optimality result on h from [25]. Under **IRM** (**Independent Reference Model**), which is a common assumption in the field [2, 15, 42], we obtain that constant TTL for a given $E[T]$ results in highest freshness for all N_U. Given a link between TTL and capacity-based caches [25, 27, 38], this result establishes that FIFO eviction achieves higher freshness than Random. In prior work, which modeled only h, these strategies were considered equivalent. We then extend our results to *proactive* caching, where the resolver pulls records from the source as soon as they expire, and quantify the reduction in freshness this results in.

Armed with a general expression for f, our second contribution is to present a framework for measuring this value at the cache, which allows it to passively monitor staleness and vary T in order to achieve the desired objectives. Because f requires the residual distribution of inter-update delays, this problem maps to method M_6 in *comparison-based blind sampling* [35]. However, M_6 has quadratic complexity in the number of samples and is generally slow to converge. To overcome this problem, we propose a method called PASS that leverages upper/lower bounds on the update age using a concave, non-parametric **EM** (**Expectation Maximization**) estimator for interval-censored data. We then show that our approach has better accuracy and much faster computation time than M_6, making it ideal for blind sampling under random inter-download delays.

A more challenging scenario arises when f must be estimated using client queries to the cache, which allows not only researchers, but also CDN companies (e.g., Akamai), to monitor freshness of relevant records at arbitrary local resolvers without having access to cache logs. We call this framework *active* and assume that the observer can issue iterative queries to the local DNS server that do not trigger downloads from the source. The problem reduces to passive sampling if T is constant; however, the situation is considerably more complicated under random TTL because the client is unaware of the download instances from the source and thus cannot apply any of the previous methods. To overcome this, our third contribution is to propose a method we call ACT_1 that estimates the update distribution without the knowledge of the download process or the distribution of TTL. While ACT_1 works reasonably well and is low-overhead, it does not come anywhere near PASS in terms of accuracy during estimation of f. Consequently, our fourth contribution is to design a method we call ACT_2 that employs the recovered $F_T(x)$ to probabilistically bound the download points of the cache. We find that the resulting technique is highly accurate and explain how the proposed system can additionally produce the remaining parameters of the system – inter-update distribution $F_U(x)$, hit rate h, and client-request rate λ – with no extra overhead. In previous work [2, 15, 42], none of these metrics were available under random T. We finish the article with Internet experiments that highlight the performance of our models and techniques.

Compared to the conference version [55], this article presents novel theoretical results throughput Section 3, a more detailed and accessible explanation of the models, a new sampling technique and overhead analysis in Section 5, all missing proofs, and numerous additional experimental results in Section 6.

2 Related Work

The first direction in previous studies focuses on modeling TTL-based caching, where hit rate h has been the most common metric of interest. The majority of literature [2, 15, 42] assumes

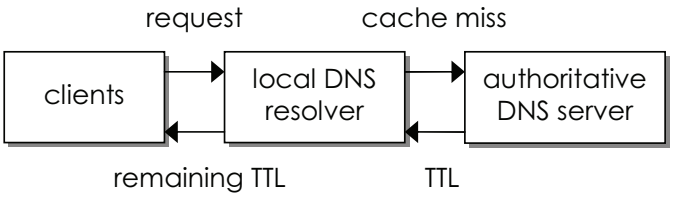


Fig. 1. Operation of DNS caching.

that clients send background queries using a Poisson process with some rate λ, which in certain cases approximates user requests rather well [16]. General renewal processes are considered in [25, 31]; however, they do not admit closed-form expressions for h and require numerical solutions to the renewal equation. The TTL is usually modeled as a constant [2, 31, 42], although h has been analyzed under a more general distribution in [5, 25, 31]. In all cases except [5], the time to download the object is assumed negligible compared to the TTL.

The second direction aims to remotely sample DNS resolvers and obtain an estimate of λ, assuming the cache applies constant TTLs known to the observer. This usually requires either Poisson queries [2, 42] or hyper-exponential inter-request delays [37]. For arrivals with longer memory, it currently appears difficult to recover λ with high accuracy.

The last direction studies object staleness under networked replication. The interplay between general update/download processes is covered in [34] and a number of unbiased techniques for estimating the update distribution is proposed in [35]. We review some of their results below. In the context of DNS, freshness is taken into account only by [15], which analyzes the expected number of missed updates between a download and subsequent queries. If updates and user queries are both Poisson, [15] proposes a staleness-measurement technique that requires sources and their caches to exchange real-time information on the observed updates/requests. Because we neither assume changes to the DNS protocol nor expect sources to cooperate with the measurement process, this is an orthogonal problem to the one studied here.

3 Performance Metrics of DNS Caches

This section introduces the terminology used later in the article, puts our problem in the context of prior literature, and provides intuition behind the staleness/efficiency tradeoff.

3.1 System Operation and Notation

Assume a system with a single source, a single replica, and a number of clients that query the replica for a particular data record/object owned by the source. As shown in Figure 1, a common networking scenario covered by this model is DNS, where the source is an authoritative server for some domain, the replica is a local DNS resolver (i.e., cache), and clients are regular end-hosts/users. The replica operates based on the TTL supplied by the source—each download $k = 1, 2, \ldots$ is accompanied by a parameter $T_k \geq 0$ that specifies how long the item can remain cached. When the cache replies to clients, it provides the *residual TTL*, which is the remaining delay before the record must be discarded. This information is valuable not only during hierarchical caching (e.g., within the OS or browser), but also in remote measurement, as we discuss shortly.

In normal DNS operation, the source decides $\{T_k\}$ using some internal mechanism (e.g., based on the current load on the available servers, routing preferences, record volatility), which generally makes $\{T_k\}$ a time-varying process. On top of this, caches may alter the source-provided TTL to satisfy their own objectives. Existing studies [4, 10, 41, 45] show that a large fraction of DNS resolvers violate the source-provided TTL, where the reasons for such behavior include attempts

to impede cross-site scripting attacks [30], defense against DDoS attacks [39], load balancing [12–14, 22], and reduction of cache inconsistency through TTL adaptation [15]. Therefore, the *effective TTL* (i.e., the one actually used by the replica) may be highly variable not only due to source decisions, but also cache policies. To cover such cases, we depart from the common assumption of constant TTL and model set $\{T_k\}$ as iid random variables with some distribution $F_T(x)$. Note that most of the convergence proofs in the article hold under more general (non-iid) conditions on $\{T_k\}$, which we briefly discuss below.

At the source, suppose the record sustains the ith update at time u_i and process $N_U(t) = \max(i : u_i \leq t)$ counts the number of such events in $[0, t]$. We assume that N_U is *age-measurable*, which is the weakest set of conditions, defined in [34], under which the various sample-path averages related to staleness are convergent. Age-measurability is a generalization of renewal processes that allows non-stationary dynamics as long as the empirical distribution of cycle lengths converges to a deterministic function. More formally, consider a point process N with cycle lengths $\{X_i\}_{i=1}^{\infty}$, where each $X_i \sim F_i(x)$ is a random variable. Let $\mathbf{1}_A$ be an indicator variable of event A and $\bar{F}(x) = 1 - F(x)$ be the complementary **CDF** (**cumulative distribution function**) of $F(x)$.

Definition 1 ([34]). A process N is called *age-measurable* if:

(1) For all $x \geq 0$, except possibly points of discontinuity of the limit, sample-path distribution of variables $\{X_1, \ldots, X_n\}$ converges in probability as $n \to \infty$ to some function $F(x)$:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{X_i \leq x} \xrightarrow{\text{P}} F(x); \tag{1}$$

(2) Function $F(x)$ is deterministic with mean $0 < \delta < \infty$;

(3) The average cycle length converges to δ in probability as $n \to \infty$:

$$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{\text{P}} \delta = \int_0^{\infty} \bar{F}(x)dx. \tag{2}$$

Defining $U_i = u_{i+1} - u_i$ to be inter-update delays, age-measurability of N_U implies that the collection of variables $\{U_i\}_{i=1}^{\infty}$ has some distribution, which we call $F_U(x)$. Similarly, denote by q_j the time of client query $j = 1, 2, \ldots$ and let inter-query intervals $Q_j = q_{j+1} - q_j$ be iid random variables with some distribution $F_Q(x)$. Suppose $N_Q(t) = \max(j : q_j \leq t)$ is the corresponding renewal process, which we assume has no point at zero, i.e., $q_1 \sim F_Q(x)$. Now observe that a combination of N_Q and $F_T(x)$ uniquely defines the *download process* N_D between the source and the cache, whose kth point is $d_k = \min(q_i : q_i > d_{k-1} + T_{k-1})$, where $d_1 = q_1$, and $N_D(t) = \max(k : d_k \leq t)$. Following Figure 2, $D_k = d_{k+1} - d_k$ represents inter-download gaps of N_D, dashed arrows are synchronization instances with the source, and the bold-line ON/OFF process N_T corresponds to object presence in the cache, i.e., $N_T(t) = 1$ when $t \in [d_k, d_k + T_k]$ and $N_T(t) = 0$ otherwise. For all cycles lengths, we assume their means are positive and finite. The rest of the section demonstrates how to use the introduced variables to model performance of DNS caches.

3.2 Hit Rate

Define $T \sim F_T(x)$ to be a generic variable with the same distribution as the TTL. Recalling that N_Q is renewal, hit rate [5, 25, 31]

$$h = \frac{E[N_Q(T)]}{1 + E[N_Q(T)]} \tag{3}$$

is the fraction of queries that arrive during ON periods in Figure 2. The numerator of (3) is the expected number of client queries in $[0, T]$, i.e., after the record has been cached, and the

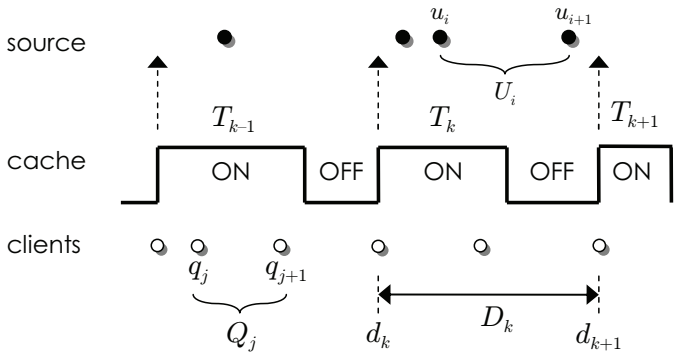


Fig. 2. Process notation.

denominator adds 1 to account for the first query that misses the cache and starts each ON cycle. Analysis of (3) is far from straightforward since it involves the renewal function $r(t) := E[N_Q(t)]$. In general, $r(t)$ can be represented as either an infinite convolution or a solution to the renewal equation [54]

$$r(t) = F_Q(t) + \int_0^t r(t-x)dF_Q(x). \tag{4}$$

Despite these complexities, there are several observations that can be made about h. First, for certain classes of $F_Q(x)$, we can determine the best distribution $F_T(x)$ that maximizes h for a given mean $E[T]$. Although [25, Proposition 4] states that constant T is optimal for all concave $F_Q(x)$, there is a flaw in that proof. The authors differentiate (4) twice; however, the second application of Leibniz's integral rule is missing a positive term $F'_Q(0)r'(t)$, which renders the conclusion invalid. Instead, we offer a different sufficient condition.

Recall that a non-negative variable X is called ***Decreasing Failure Rate*** (**DFR**) if $P(X > x + y|X > y)$ is increasing in y for each $x > 0$ [9]. Note that DFR distributions are commonly found in practice (e.g., all heavy-tailed cases such as Pareto and Weibull).

Theorem 3.1. *Assume $E[T]$ is fixed. If $F_Q(x)$ is DFR, then constant T maximizes* (3).

Proof. It is well-known that $r(t)$ is concave [9] if renewal cycles are DFR (although the opposite is false [56]). From Jensen's inequality, we get $E[r(T)] \le r(E[T])$. Therefore,

$$h = \frac{E[r(T)]}{1 + E[r(T)]} \le \frac{r(E[T])}{1 + r(E[T])}. \tag{5}$$

As a result, replacing a random T with a constant equal to its mean allows h to achieve the upper bound in (5), i.e., yields the largest possible hit rate. □

Second, notice that $E[N_Q(T)]$ increases for stochastically larger T or stochastically smaller Q, which implies that h in (3) does too. Intuitively, this makes sense – longer ON periods in Figure 2 or faster query rates yield better hit rates. The opposite holds when the conditions are reversed.

Third, it should be noted that (3) holds even if the TTLs are non-iid. In particular, if the point process defined by cycle lengths $\{T_k\}$ is age-measurable with a limiting distribution $F_T(x)$, we get from Vitali's Convergence Theorem [17], bounded nature of renewal functions (i.e., $0 \le r(t) \le a + bt$ for some a, b), and uniform integrability that

$$\frac{1}{n}\sum_{i=1}^{n} r(T_i) \xrightarrow{\text{P}} \int_0^\infty r(x)dF_T(x), \tag{6}$$

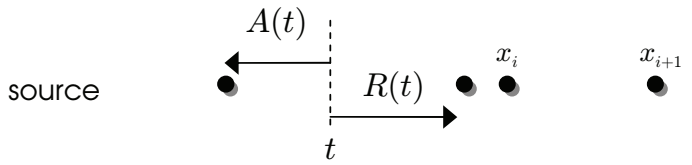


Fig. 3. Age and residual of a point process.

from which it follows that the sample-path averaged hit-rate h converges in probability to the result in (3). Similar extensions to non-iid $\{T_k\}$ apply later in this section, but for brevity we omit discussing them explicitly.

3.3 Freshness

As the DNS record keeps changing, cache responses may become outdated compared to the current state at the source. For the example in Figure 2, the record provided by the cache to clients at time q_j is fresh (i.e., the same as at the source), while that at q_{j+1} is stale due to an update in $(q_j, q_{j+1}]$. Thus, focusing solely on maximization of h, which can be achieved by setting $T_k = \infty$, is often meaningless in practice. Instead, TTL-based systems need to balance between hit rates and freshness, which we explore in more detail below.

To avoid repetitive definitions, assume a generic point process N with events at $\{x_1, x_2, \ldots\}$ and cycle lengths $X_i = x_{i+1} - x_i$. Then, following the illustration in Figure 3, let the *age* of N at time t be the delay to the previous point

$$A(t) := t - x_{N(t)} \tag{7}$$

and the *residual* be the distance to the next point

$$R(t) := x_{N(t)+1} - t. \tag{8}$$

If N is age-measurable, the distributions of variables (7)–(8) sampled at uniform points t converges in probability to the equilibrium CDF of $F(x)$ [34, Theorem 1]

$$G(x) := \frac{1}{\delta} \int_0^x \bar{F}(y)dy, \tag{9}$$

where $F(x)$ is the limit from (1) and $\delta > 0$ is its mean. It is then convenient to use random variable $A \sim G(x)$ to represent the age of distribution $F(x)$ and $R \sim G(x)$ to indicate its residual. When applying these definitions to N_U, N_Q, N_D, the notation in (7)–(9) is augmented with a corresponding subscript $\{U, Q, D\}$. For example, the update process N_U yields $A_U(t), R_U(t), G_U(x)$, and $A_U, R_U \sim G_U(x)$. On the other hand, since N_T is a binary ON/OFF process, (7)–(9) do not directly apply to it. Instead, we define the age/residual of N_T only for the ON periods, which translates into

$$A_T(t) := \begin{cases} t - d_{N_D(t)} & N_T(t) = 1 \\ \text{undefined} & \text{otherwise} \end{cases}, \tag{10}$$

$$R_T(t) := \begin{cases} d_{N_D(t)} + T_{N_D(t)} - t & N_T(t) = 1 \\ \text{undefined} & \text{otherwise} \end{cases}, \tag{11}$$

$$G_T(x) := \frac{1}{E[T]} \int_0^x \bar{F}_T(y)dy, \tag{12}$$

and $A_T, R_T \sim G_T(x)$.

Now suppose *freshness* f is the long-term fraction of queries that return a record that is consistent with that at the source. It is also common to use the term *staleness* [40, 53] to refer to $1 - f$. For either quantity to be computable using the properties of the underlying processes (i.e., without access to both source and cache logs), it is necessary that processes (N_U, N_Q) be *age-independent* [34], which means that their cycle lengths not enter a permanent phase-lock as $t \to \infty$. In more detail, let W_t be a uniform random variable in $[0, t]$ and consider the following.

Definition 2 ([34]). Two age-measurable point processes N_U and N_Q are called *age-independent* if for all $x, y \geq 0$:

$$\lim_{t\to\infty} P(A_Q(W_t) < x | A_U(W_t) = y) = G_Q(x). \tag{13}$$

This condition can be satisfied by requiring that one of $F_U(x), F_Q(x)$ be non-lattice (i.e., not defined on rational numbers). For the remainder of the article, we assume that age-independence of (N_U, N_Q) holds.

Theorem 3.2. *Cache replies are fresh with probability*

$$f = \frac{1 + E[N_Q(\min(R_U, T))]}{1 + E[N_Q(T)]}. \tag{14}$$

Proof. For a download cycle k in Figure 2, there are $N_Q(T_k) + 1$ total queries, the first of which at time d_k is always fresh. Since we know $R_U(d_k)$ is the amount of time before the next update, it follows that $1 + N_Q(\min(R_U(d_k), T_k))$ initial requests in each cycle k encounter a fresh response. Therefore, the fraction of non-stale replies is given by

$$f = \lim_{n\to\infty} \frac{\sum_{k=1}^{n}[1 + N_Q(\min(R_U(d_k), T_k))]}{\sum_{k=1}^{n}[1 + N_Q(T_k)]}. \tag{15}$$

If (N_U, N_Q) are age-independent, so are (N_U, N_D) because N_Q is renewal and N_D is a thinned version of N_Q. Using [34], this means that the distribution of residuals $\{R_U(d_k)\}$ observed by the download process converges to (9). Combining this with the iid nature of $\{T_k\}$, we get (14). □

3.4 Freshness-Efficiency Tradeoff

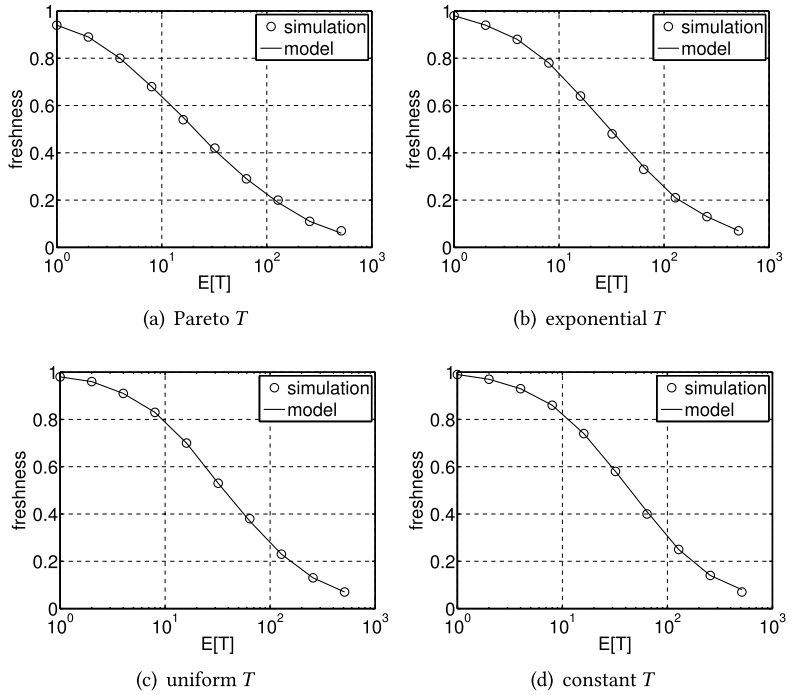
The effect of T and Q on freshness is quite a bit more complex than on the hit rate. Simply making T stochastically larger (or Q smaller) is not enough to predict the change in f in every circumstance. The problem is that (14), unlike (3), has different expressions in the numerator and denominator. In one special case, however, we can establish the asymptotics of how the TTL and query rates affect staleness.

Theorem 3.3. *Let $\{F_T^n(x)\}$ be a sequence of TTL distributions for $n = 1, 2, \ldots$ and define $T^n \sim F_T^n(x)$. Assuming $E[T^n] \to \infty$ and $F_Q(x)$ remains fixed as $n \to \infty$, freshness under T^n converges to 0. Similarly, if $Q^n = Q/n$, where $Q \sim F_Q(x)$, and $F_T(x)$ is fixed as $n \to \infty$, freshness under a query process driven by cycle lengths Q^n converges from above to $E[\min(R_U, T)]/E[T]$ as $n \to \infty$.*

Proof. Suppose $E[T^n] \to \infty$ and the distribution of Q is given. Then, from the Elementary Renewal Theorem [54] and $E[Q] < \infty$, it follows that $E[N_Q(T^n)] \to \infty$. Since $E[R_U] < \infty$, we get that $E[N_Q(\min(R_U, T^n))]$ is upper-bounded by $E[N_Q(R_U)] < \infty$ and thus the ratio in (14) diminishes to zero.

Now suppose $Q^n = Q/n$, where $Q \sim F_Q(x)$ is such that $0 < E[Q] < \infty$, and $F_T(x)$ is given. Notice that $N_{Q^n}(t)$ has the same distribution as $N_Q(nt)$. Defining random variable $X = T/E[Q]$, observe from the Elementary Renewal Theorem that

$$X_n := \frac{r(nT)}{n} \to X, \tag{16}$$



(a) Pareto T (b) exponential T

(c) uniform T (d) constant T

Fig. 4. Simulations under Pareto U and Q, where $E[U] = 20$ sec and $\lambda = 1$/sec (observation window with 1M samples).

where as before $r(t) = E[N_Q(t)]$. Since $r(t)/t \to 1/E[Q] < \infty$, we get that $r(t) \le at + b$ for some constants (a, b). This means X_n is upper-bounded by $aT + b/n$ with probability 1. Therefore, setting $Y = aT + b$ yields that $X_n \le Y$ for all n. Because $E[Y] < \infty$, the Dominated Convergence Theorem [43] produces $E[X_n] \to E[X]$, or equivalently

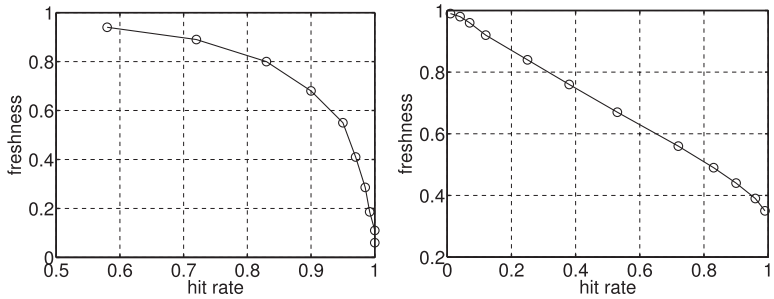
$$\frac{E[r(nT)]}{n} \to \frac{E[T]}{E[Q]}, \tag{17}$$

which implies that (14) converges to its lowest point $E[\min(R_U, T)]/E[T]$ as $n \to \infty$. □

Intuitively, this result holds because larger TTLs yield less-frequent downloads from the source and larger query rates λ produce more stale replies per synchronization event. With the exception of esoteric counter-examples, this generally means that *freshness and hit rate are tradeoffs of each other.*

3.5 Discussion

To show this better, we next examine several scenarios. Figure 4 shows a comparison between simulations and model (14) under a general N_Q with a mean 20-sec inter-update delay. Notice that the two match very well and that freshness indeed decreases as $E[T]$ gets larger. In Figure 5(a), we vary $E[T]$ and plot f as a function of h, where the two models come from (3) and (14). As predicted, longer TTLs drive $h \to 1$ and $f \to 0$, reaffirming the tradeoff. A similar picture emerges in Figure 5(b) as λ changes, except here $f \to E[\min(R_U, T)]/E[T] = 1/3$ as hit rate $h \to 1$.



(a) changing $E[T]$ ($\lambda = 1$) (b) changing λ ($E[T] = 40$)

Fig. 5. Cache tradeoffs (Pareto U, T, Q with $E[U] = 20$ sec).

3.6 Large User Base

When the local resolver sustains queries from a large population of users, it might be sensible to model N_Q as a Poisson process, which is a common assumption in the field [2, 15, 42]. Under exponential Q, inter-download delay D becomes a convolution of ON/OFF cycle lengths, i.e., $D = T + Q$. Furthermore, letting $\lambda = 1/E[Q]$ be the request-arrival rate, (3) transforms into

$$h = \frac{\lambda E[T]}{1 + \lambda E[T]} \tag{18}$$

and (14) simplifies to

$$f = \frac{1 + \lambda E[\min(R_U, T)]}{1 + \lambda E[T]}. \tag{19}$$

The next result establishes a more useful representation of freshness by making it an explicit function of h.

Theorem 3.4. *Under Poisson queries, freshness is given by*

$$f = 1 - h + hp, \tag{20}$$

where $p := P(R_T < R_U)$.

Proof. Notice that

$$E[\min(R_U, T)] = \int_0^\infty P(\min(R_U, T) > z)dz = \int_0^\infty P(R_U > z)P(T > z)dz. \tag{21}$$

Since $G_T(x)$ is the CDF of residuals for $F_T(x)$, its density $g_T(x) := (1 - F_T(x))/E[T]$ exists. We then get that $P(T > z) = E[T]g_T(x)$ and

$$E[\min(R_U, T)] = E[T] \int_0^\infty (1 - G_U(z))g_T(z)dz = E[T]P(R_U > R_T). \tag{22}$$

Substituting (22) into (19) gives:

$$f = \frac{1 + p\lambda E[T]}{1 + \lambda E[T]} = \frac{1}{1 + \lambda E[T]} + \frac{p\lambda E[T]}{1 + \lambda E[T]}, \tag{23}$$

which transforms into (20) using (18). □

There is a simple explanation for (20). The first term $1 - h$ covers queries that miss the cache, in which case they are fresh with probability 1. The second term hp applies to queries that hit an ON interval in Figure 2, in which case age-independence between (N_U, N_Q) and the results in [34]

ensure that content is fresh with probability p. Next, re-writing (20) as $f = 1 - h(1-p)$, it is clear that increasing λ, which also increases h, causes freshness to go down. This is consistent with our earlier conclusions about (14). On the other hand, the impact of T is less obvious because h and $1-p$ may move in different directions.

Theorem 3.5. *For Poisson queries with a fixed rate λ, increasing $E[T]$ such that residual R_T becomes stochastically larger reduces freshness.*

Proof. Stochastically larger residuals R_T makes $p = P(R_T < R_U)$ smaller. At the same time, increasing $E[T]$ scales h up. Therefore, $f = 1 - h(1-p)$ must decrease. □

One simple strategy to grow $E[T]$ and stochastically increase R_T is to use *scaling*, i.e., replace T with ρT, where $\rho > 1$. It should be noted that ρT has the same distribution as T, but with a different mean. For example, $T \sim \exp(\lambda)$ becomes $\exp(\lambda/\rho)$, uniform$[a, b]$ shifts to uniform$[\rho a, \rho b]$, and Pareto(α, β) transforms into Pareto$(\alpha, \rho\beta)$.

3.7 Relationship to Other Strategies

There is an interesting parallel between TTL-based caches studied here and two capacity-based strategies—Random and FIFO. Existing work [25, 27, 38] shows that these methods can be approximated by a TTL-based cache, where Random uses exponential T and FIFO relies on constant T. These policies have the same hit rate in (18); however, they are clearly different when freshness is taken into account. The next result proves that FIFO offers better freshness than Random under all update processes N_U.

Theorem 3.6. *Assume Poisson queries. For a given $E[T]$ and $F_U(x)$, constant TTL delays achieve the largest freshness. For a given $F_T(x)$ and $E[U]$, constant update delays produce the worst freshness.*

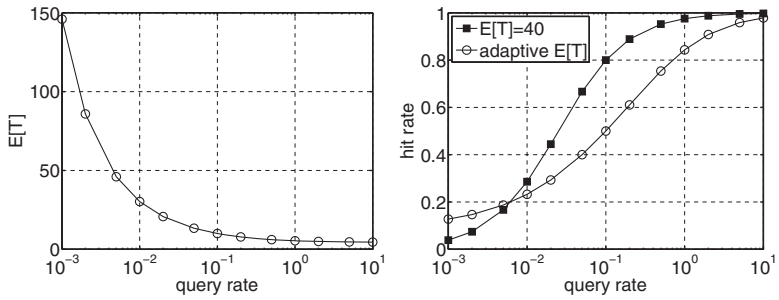
Proof. For a fixed $E[T]$, the hit rate remains constant, which means that freshness in (20) is determined solely by $p = P(R_T < R_U)$. Suppose $F_U(x)$ and $E[T]$ are fixed. Then, [34] shows that p is maximized when T is a constant. Similarly, assume $F_T(x)$ and $E[U]$ are given. Then, p is minimized when U is a constant [34]. □

In order to reduce response latency, other studies [15, 16, 21] propose that the cache pre-fetch records from the server as soon as the TTL expires. In this technique, which is called *proactive caching*, hit rate h is always 1 and thus $f = p$ from (20). Compared to regular (non-proactive) operation, where $h < 1$ and $f \geq p$, this is another example of the efficiency-staleness tradeoff – h becomes maximally optimal, but f is maximally suboptimal. Furthermore, the download rate from the source increases from $1/E[D] = 1/E[T+Q]$ to $1/E[T]$. This indicates that proactive caching improves the hit rate (i.e., latency) at the cost of higher communication bandwidth and lower consistency of records served to clients. The models developed in this section can be used to quantify this tradeoff.

4 Passive Measurement

4.1 Preliminaries

Assume that the local resolver needs to estimate the long-term freshness f of a given record in its cache. The reasons for this objective could be numerous (e.g., performance monitoring, source characterization), but consider a more concrete example. Suppose the source provides $T_k = 40$ sec for all k, but this leads to 10% freshness for the specific user process N_Q at this cache. Assuming that the replica has enough spare bandwidth to sustain more frequent downloads, it may decide to lower the TTL (i.e., preemptively evict records that are likely stale) in order to achieve a certain freshness guarantee to its clients, while still maintaining a reasonable level of latency. Since process



(a) adaptive TTL (b) resulting hit rate

Fig. 6. Achieving 90% freshness (Pareto U, T, Q with $E[U] = 20$ sec).

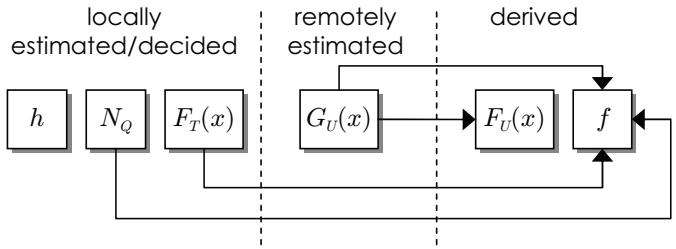


Fig. 7. Model roadmap for passive monitoring.

N_Q is localized to each cache, the source has no ability to optimize $\{T_k\}$ simultaneously for all of its replicas. Thus, one feasible application that needs f, which we call an *adaptive* (i.e., staleness-aware) cache, selects $G_T(x)$ such that freshness f in (14) is maintained at some desired threshold. To see how this works, consider the scenario in Figure 5, deterministic T, and 90% freshness. For this setup, Figure 6(a) shows the relationship between the query rate λ and TTL. Observe that more-frequent client requests mandate a sharp decrease in $E[T]$, while less-loaded conditions do the opposite. The corresponding hit rate is provided in Figure 6(b) in comparison to the default 40-sec eviction delay. If the record is requested more often than once every 2 min, the adaptive strategy exhibits lower hit rates and requires more bandwidth, but provides fresher records than the non-adaptive version. The situation is reversed when the record is less popular.

Beyond DNS, passive sampling arises in the context of web crawlers, where the inter-update distribution $F_U(x)$ is needed to schedule future downloads in order to balance between bandwidth and freshness. Previous work [34, 35] deals with this particular case in more detail, but there are numerous other applications on the Internet with similar requirements (i.e., anything that has an updating source and a replica, such as CDNs).

As shown in Figure 7, parameters h, N_Q, and $F_T(x)$ can be locally determined by the resolver. On the other hand, estimation of f at the replica, which we call *passive sampling*, requires an inference process that obtains the residual update distribution $G_U(x)$. The rest of the section focuses on this.

4.2 Sampling Update Age

Recall from Figure 2 that the cache contacts the source at points $\{d_k\}$, which form some download process N_D. From our assumptions, (N_U, N_D) are age-independent, which allows application of the various techniques in blind sampling [34]. There are three variations of methods based on the capabilities of the source. In the first case, the source explicitly provides the update age $A_U(d_k)$

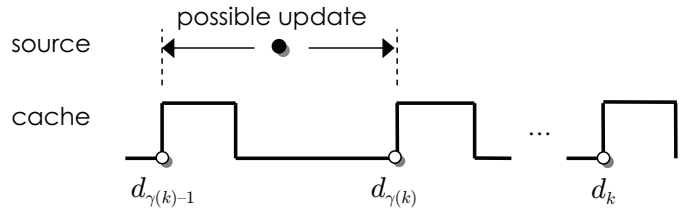


Fig. 8. Bounding the update age in passive measurement.

from Figure 3 during each download k, e.g., using protocol fields such as HTTP headers. For this scenario, [35] develops a method called M_2 that is asymptotically unbiased and quickly convergent. Unfortunately, this scenario does not apply to DNS. As a result, the cache has to infer presence of updates by comparing adjacent versions of downloaded records. Define a binary process Δ_k to be 1 if the object is detected as modified during download k. When inter-download delays D_k are all constant, the second class of methods in [34] uses $\{\Delta_k\}_{k=1}^{\infty}$ to estimate $G_U(x)$ with asymptotically consistency. However, again, this formulation does not work for our problem since D_k is a complex random variable that subsumes T_k and the preceding OFF period in Figure 2, both of which are random.

This leaves us with the third class of methods in [35], which consists of a single technique M_6 that works with random D_k and reconstructs $G_U(x)$ from observations $\{\Delta_k\}_{k=1}^{\infty}$. It has two drawbacks—quadratic computation time in the number of downloads n and lower accuracy compared to the remaining methods in [35]. Therefore, our first goal is to improve this technique in both aspects.

4.3 Non-Parametric EM

Blind-sampling methods work by estimating the unknown update age $A_U(d_k)$ in download points. Instead of rounding this to some value, as done in [34], our novel approach is to provide the estimator with *interval-censored* values, i.e., upper/lower bounds on $A_U(d_k)$. Define $\gamma(k) = \max(i \leq k : \Delta_i = 1)$ to be the last download in $[0, d_k]$ that detected a modification. Consider Figure 8 and suppose the current time is d_k. Then, the latest update at the source is always confined to the interval $(d_{\gamma(k)-1}, d_{\gamma(k)}]$. This immediately yields

$$d_k - d_{\gamma(k)} \leq A_U(d_k) \leq d_k - d_{\gamma(k)-1}. \tag{24}$$

Suppose the lower bound in (24) is L_k and the upper is R_k. Then, our technique, which we call PASS, collects a sequence of pairs $\{(L_k, R_k)\}_{k=1}^{n}$ and feeds them into a non-parametric EM estimator. Our model draws inspiration from Turnbull's method for censored intervals [51]. First, we quantize the bounds to be a multiple of some bin size, where L_k is rounded *down* and R_k *up*. We then combine upper/lower bounds into a single vector, sort the result ascending, eliminate duplicates, and obtain a new set $(x_1, \ldots, x_m)$, where $m \leq 2n$. From this, we can form non-overlapping bins $B_i = [x_{i-1}, x_i)$, where $x_0 = 0$.

Let $p_i(t)$ be the estimated probability that the target random variable $A_U \sim G_U(x)$ belongs to bin B_i during step t of the iteration, where $p_i(0) = 1/m$ for all i. Now define a_{ik} to be an indicator variable that B_i is entirely contained in the discretized interval from download k. Using non-quantized bounds, this can be expressed as

$$a_{ik} = \begin{cases} 1 & B_i \cap [L_k, R_k] \neq \emptyset \\ 0 & \text{otherwise} \end{cases}. \tag{25}$$

ALGORITHM 1: PASS's implementation of (27).

```
1  Function FastEM(intervals, m)
2      p = (1/m, 1/m, ..., 1/m);                          ◃ initial guess
3      while not converged do
4          p = OneIteration (intervals, p, m)
5  Function OneIteration(intervals, p, m)
6      Z = zeros (1, m+1);                                ◃ temp storage
7      cdf = prefix_sum ([0 p]);                          ◃ CDF padded with 0 at front
8      for k = 1 to intervals.size() do
9          Vk = cdf [intervals [k].R] − cdf [intervals [k].L];
10         Z [intervals [k].L] += intervals [k].count / Vk;
11         Z [intervals [k].R] −= intervals [k].count / Vk;
12     psum = 0;                                          ◃ prefix sum of Z
13     for i = 1 to m do
14         psum += Z [i];                                 ◃ total weight from all intervals
15         p [i] ∗= psum / n ;                            ◃ normalize and save
16     return p;
```

Next, define $V_k(t)$ to be the probability that $A_U \sim G_U(x)$ belongs to $[L_k, R_k]$ during iteration t

$$V_k(t) = \sum_{i=1}^{m} a_{ik} p_i(t). \tag{26}$$

Each probability is then refined using recurrence

$$p_i(t+1) = \frac{p_i(t)}{n} \sum_{k=1}^{n} \frac{a_{ik}}{V_k(t)} \tag{27}$$

until the stopping criterion is satisfied, i.e., $||p(t+1) - p(t)|| < \epsilon$, where $\epsilon > 0$ is a constant, and $p(t) = (p_1(t), \ldots, p_m(t))$. Note that this process is asymptotically accurate [51], i.e., (27) converges to $G_U(x)$ as $n \to \infty$.

4.4 Implementation

A naive version of (25)–(27) calculates all mn values a_{ik} and keeps them in RAM, which is highly inefficient. In the worst case (i.e., $m = 2n$), this computation is quadratic in both space/time. Instead, we offer Algorithm 1 whose per-iteration CPU complexity $O(n)$ and storage cost $O(m)$ are optimal. Prior to calling FastEM, assume the program has already determined bin boundaries $(x_1, \ldots, x_m)$ and mapped each pair (L_k, R_k) to the appropriate bin using an array of structs, i.e., `intervals[k].L` and `intervals[k].R`. Note that duplicate tuples (L_k, R_k) are compressed such that `intervals[k].count` is the corresponding frequency.

After initialization, Algorithm 1 computes the CDF of A_U using a prefix sum of the **PMF** (**probability mass function**) array p (Line 7). Padding with a front zero is needed to properly compute V_k in Line 9. We then use a temporary array Z to accumulate all weights $1/V_k$ that will be distributed into the relevant bins after the loop is over. Specifically, $Z[i]$ stores increments that must be applied to the PMF in position $[i, m]$. This requires adding $1/V_k$ at the left boundary of the interval (Line 10) and subtracting it at the right boundary (Line 11). The second loop in Lines 13-15 computes a prefix sum of Z and stores the result, normalized by $p_i(t)/n$, into the same vector.

With compression of duplicate intervals, the number of unique boundaries supplied to Algorithm 1 is upper-bounded by $\min(m(m-1)/2, n)$. If $m \ll n$ and the runtime is dominated by

iteration in Lines 3-4, rather than the initial $n \log n$ sort, PASS can exhibit sublinear scaling in a limited range of n. We show such an example below.

4.5 Concave EM

Note that $G_U(x)$ in (9) has a monotonically decreasing density $g_U(x) = G'_U(x) \sim 1 - F_U(x)$. As a result, $G_U(x)$ is a concave function. This is an important property that must be preserved by the estimator, especially if recovery of $F_U(x) = 1 - g_U(x)/g_U(0)$ is needed from $G_U(x)$. Since the density is commonly estimated by scaling and smoothing the PMF, a proper solution must guarantee $p_i(t) \geq p_{i+1}(t)$. Unfortunately, (25)–(27) fail to do so. Furthermore, none of the previous literature has considered this issue before.

To overcome the setback, we offer a new EM algorithm that ensures proper recovery of residual (i.e., concave) CDFs. Define $\delta_i = x_i - x_{i-1}$ to be the length of the ith bin and let $g_i(t) = p_i(t)/\delta_i$ be the corresponding estimate of density, where $g_{m+1} = 0$. Suppose

$$q_i(t) = x_i(g_i(t) - g_{i+1}(t)) \tag{28}$$

models the negative derivative of $g_i(t)$, normalized such that $\sum_{i=1}^{m} q_i(t) = 1$. Observe that if the estimated density $g_i(t)$ is a decreasing function of i, then $q_i(t) \geq 0$ for all i. The opposite holds as well since

$$g_i(t) = \sum_{j=i}^{m} \frac{q_j(t)}{x_j}. \tag{29}$$

We next create an EM algorithm for $q_i(t)$ such that $q_i(t) \geq 0$ is preserved during each iteration.

Theorem 4.1. *A concave EM estimator for interval-censored data is given by*

$$q_i(t+1) = \frac{q_i(t)}{nx_i} \sum_{j=1}^{i} \delta_j \sum_{k=1}^{n} \frac{a_{jk}}{V_k(t)}. \tag{30}$$

Proof. Conditioning on all intervals $[L_k, R_k]$, we have

$$P(A_U \in [L_k, R_k]) = \sum_{i=1}^{m} a_{ik}(G_U(x_i) - G_U(x_{i-1})). \tag{31}$$

Letting p_i be an estimate for $G_U(x_i) - G_U(x_{i-1})$ such that $\sum_{i=1}^{m} p_i = 1$, the objective is to maximize the likelihood function $\mathcal{L}(\mathbf{p})$ with respect to vector $\mathbf{p} = (p_1, \ldots, p_m)$, i.e., the probability of observing a particular sequence of intervals, where

$$\mathcal{L}(\mathbf{p}) = \prod_{k=1}^{n} P(A_U(d_k) \in [L_k, R_k]) = \prod_{k=1}^{n} \sum_{i=1}^{m} a_{ik} p_i, \tag{32}$$

or equivalently

$$\log(\mathcal{L}(\mathbf{p})) = \sum_{k=1}^{n} \log\left(\sum_{i=1}^{m} a_{ik} p_i\right), \tag{33}$$

subject to the constraint that the estimate for G_U is concave. Even through $A_U(t)$ and $A_U(s)$ for $s > t$ technically are not independent, age-measurability of N_U and age-independency of (N_U, N_D) ensures that asymptotically the collection of variables $\{A_U(d_k)\}_{k=1}^{n}$ acts as an iid sequence [34].

Assuming the bins remain the same as $n \to \infty$,

$$\frac{1}{n} \sum_{k=1}^{n} \log\left(\sum_{i=1}^{m} a_{ik} p_i\right) \to E\left[\log\left(\sum_{i=1}^{m} \mathbf{1}_{x_{i-1} < A_U \leq x_i} p_i\right)\right] = \sum_{i=1}^{m} (G_U(x_i) - G_U(x_{i-1})) \log p_i. \tag{34}$$

The limit is uniquely maximized by

$$p_i = \frac{G_U(x_i) - G_U(x_{i-1})}{G_U(x_m)}, \tag{35}$$

which shows that optimizing the likelihood function with respect to variables $(p_1, \ldots, p_m)$ indeed produces the desired quantity. The main caveat is that we must ensure a concave estimate at every step t and for finite n. Using vector $\mathbf{q} = (q_1, \ldots, q_m)$ from (28), the objective function (33) can be re-written as

$$\log(\mathcal{L}(\mathbf{q})) = \sum_{k=1}^{n} \log V_k, \tag{36}$$

where $p_i = \delta_i g_i$ and application of (29) yield

$$V_k = \sum_{i=1}^{m} a_{ik} p_i = \sum_{i=1}^{m} a_{ik} \delta_i \sum_{j=i}^{m} \frac{q_j}{x_j}. \tag{37}$$

Since the two indexes (i, j) cover all pairs of integers such that $i \le j$, the order of sums can be switched to produce

$$V_k = \sum_{r=1}^{m} \frac{q_r}{x_r} \sum_{s=1}^{j} a_{sk} \delta_s. \tag{38}$$

Using $q_m = 1 - \sum_{i=1}^{m-1} q_i$ and differentiating (36) yields

$$l_i = \frac{\partial \log(L(\mathbf{q}))}{\partial q_i} = \frac{1}{x_i} \sum_{k=1}^{n} \frac{\sum_{s=1}^{i} a_{sk} \delta_s}{V_k} - n. \tag{39}$$

Note that the goal of the estimator is to drive each derivative l_i to zero. To accomplish this, let $q_i(t)$ be the estimate of optimal q_i during step t. From [51], the EM iteration for solving such systems is given by

$$q_i(t+1) = q_i(t)\left(1 + \frac{l_i(t)}{n}\right), \tag{40}$$

which converges to a solution (i.e., fixed point) where either $q_i(t) = 0$ or derivative $l_i(t) = 0$. As long as the initial distribution has $q_i(0) \ge 0$, we obtain that $l_i(t) \ge -n$ and thus $q_i(t)$ is guaranteed to be non-negative for all t. Using (39) and changing the order of summations one more time,

$$1 + \frac{l_i(t)}{n} = \frac{1}{n x_i} \sum_{k=1}^{n} \frac{\sum_{j=1}^{i} a_{jk} \delta_j}{V_k(t)} = \frac{1}{n x_i} \sum_{j=1}^{i} \delta_j \sum_{k=1}^{n} \frac{a_{jk}}{V_k(t)},$$

whose usage in (40) produces (30). □

Note that (30) uses variables from (25)–(26), which requires further elaboration. Before the first iteration, we set $p_i(0) = \delta_i / x_m$, which ensures a monotonic initial density, and convert $p(0)$ into vector $q(0)$ using (28). This requires one pass over all m bins. Then, we represent (30) as

$$q_i(t+1) = \frac{q_i(t)}{n x_i} \sum_{j=1}^{i} \delta_j W_j(t), \tag{41}$$

where

$$W_j(t) = \sum_{k=1}^{n} \frac{a_{jk}}{V_k(t)}. \tag{42}$$

Note that the entire set $\{W_j(t)\}_{j=1}^m$ can be computed by calling a slightly modified function OneIteration in Algorithm 1, where Line 15 does not have the $p_i(t)/n$ multiplier. Once all $\{W_j(t)\}$ are available, an extra prefix sum over m bins produces (41). Finally, $q(t+1)$ is converted back to $p(t+1)$ using (29), which requires another scan over m bins. In the end, per-iteration cost of our concave EM differs from that of Algorithm 1 by an additive term $2m$, which is negligible in practice.

5 Active Measurement

5.1 Preliminaries

We now face the issue of estimating f from a vantage point outside the cache, which we call *active sampling*. This problem may be of interest to sources (e.g., CDN companies), where the goal is to measure what types of freshness their TTL algorithms produce at certain customer networks (e.g., Comcast, Verizon). Additionally, caches may be remotely monitored by researchers and campus network administrators, who do not have access to the logs, to characterize replication efficiency and diagnose potential problems with bandwidth consumption, staleness, and performance of deployed algorithms.

Due to the lacking cooperation from the cache, computation of f is likely intractable unless N_Q is Poisson, which we assume in the rest of the article. To calculate (20), we need to estimate $p = P(R_T < R_U)$ and hit rate h, the former of which requires the residual TTL distribution $G_T(x)$ and the residual update distribution $G_U(x)$. This is illustrated in Figure 9, where $E[T]$, λ, and $F_U(x)$ can be obtained with no extra overhead once the three main parameters are known.

Note that sampling must be performed without intrusion into the ON/OFF process of the cache, which would skew the result. We therefore assume that the resolver accepts *iterative* requests, to which it responds with an error, instead of contacting the source, if the record is not currently cached. For cached objects, the resolver returns their remaining TTL, which equals $R_T(t)$ at time t using our earlier notation.

5.2 Constant TTL

Previous measurement literature [2, 37, 42] is limited to estimating λ and h under constant T. However, even in our setting that requires estimating an extra distribution $G_U(x)$, the problem becomes trivial under this condition. Assume T is a fixed value that is known to the observer from a-priori contacts with the source or other means. Checking the cache every T time units ensures that every ON period receives at least one sample point, which allows recovery of the corresponding download instance $d_k = t_k + R_T(t_k) - T$ for all query times t_k. The problem of estimating $G_U(x)$ then becomes identical to that in passive sampling, where PASS provides an excellent supporting platform. Furthermore, knowledge of the starting and ending point of each ON interval allows access to all OFF durations, i.e., $d_k - (d_{k-1} + T)$, whose average tends to $1/\lambda$. Finally, the hit rate follows from (18).

5.3 Random TTL

The issue is significantly more complex when the TTL varies between cycles, which is our assumption from this point forward. The challenge stems from the observer's uncertainty about the location of download point d_k since only the cache knows this information. This precludes direct measurement of the OFF duration or application of PASS. We now formulate our framework for solving these issues.

Suppose the observer is a special client that sends only iterative queries with an objective to determine f with asymptotic accuracy as the observation window tends to infinity. Assume that these requests are issued at points $\{s_k\}$ such that inter-sample delays $S_k = s_{k+1} - s_k$ have some distribution $F_S(x)$ and $N_S(t) = \max(k : s_k \le t)$ is an age-measurable process. Given age-independence

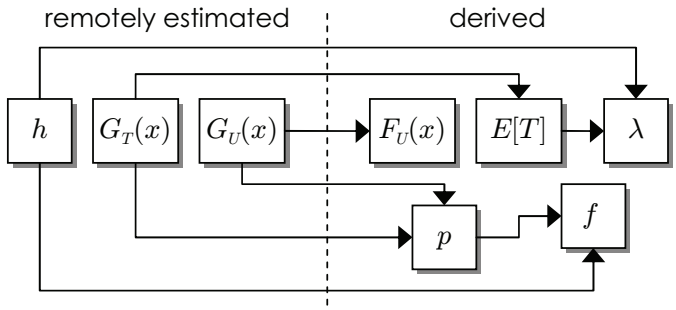


Fig. 9. Model roadmap for active monitoring.

between (N_D, N_S), the **ASTA** (**Arrivals See Time Averages**) property of the constructed system [34] ensures that the fraction of sample points that return a cached object tends to $E[T]/E[T+Q]$ as $n \to \infty$. Since this value directly equals h, i.e., the first unknown parameter in Figure 9, the observer can measure the hit rate without any additional bandwidth or computational overhead.

Leveraging [34], the observed TTL residuals $\{R_T(s_k)\}_{k=1}^{n}$ converge in distribution to $R_T \sim G_T(x)$ as $n \to \infty$. With iid $\{S_k\}$ and Poisson N_Q, reconstructing $G_T(x)$ should be possible using arbitrary distributions $F_S(x)$, including lattice cases (e.g., constant S_k). However, this analysis assumes that residuals are given by the server with high precision. In reality, the packet format of DNS requires that $R_T(s_k)$ be truncated to an integer number of seconds. To compound the issue, some servers round the residual time up (e.g., BIND), some down, and others to the nearest value (e.g., Windows IIS). Unfortunately, truncation introduces bias into the measurement and leads to poor estimation.

To overcome this problem, we offer the following approach. Suppose the local DNS server is first tested for the direction of round-offs. This can be done using three packets at time 0, 0.25, and 0.75 sec. The first packet seeds the server with a fresh TTL, while the other two differentiate between the three rounding options. Additionally, notice that if a download point s_k hits an ON interval, the true (i.e., untruncated) residual is contained in $[R_T(s_k) + a, R_T(s_k) + a + 1]$, where $a = 0$ for rounding down, $a = -1$ for rounding up, and $a = -1/2$ for rounding to the nearest integer. Furthermore, if multiple points s_k hit the same ON interval, we can take the maximum of the left bounds and the minimum of the right bounds to better pinpoint the true residual.

With this in mind, we collect one max/min bound from each sampled ON interval and pass the result through the concave-EM in (30) to yield a monotonic density $g_T(x)$. Since

$$g_T(x) = G_T'(x) = \frac{1 - F_T(x)}{E[T]}, \tag{43}$$

this procedure yields $F_T(x)$ and $E[T] = 1/g_T(0)$ using numerical differentiation of $G_T(x)$. Recalling (18), knowledge of h and $E[T]$ produces λ. Finally, assuming $G_U(x)$ is known, all remaining pieces fall into place, i.e.,

$$p = P(R_T < R_U) = \int_0^\infty (1 - G_U(x)) g_T(x) dx \tag{44}$$

and thus $f = 1 - h(1 - p)$. The only still-unknown parameter in Figure 9 is $G_U(x)$. We focus on its estimation next.

5.4 ACT_1

Our first attempt at active estimation of $G_U(x)$, which we call ACT_1, places bounds on $A_U(d_k)$ using the information available to the client and then runs iteration (30) until convergence. Note

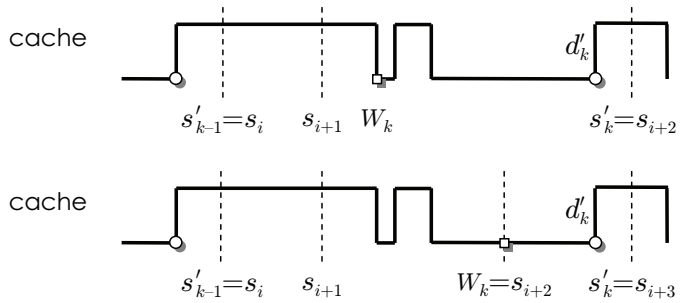


Fig. 10. Notation in active sampling.

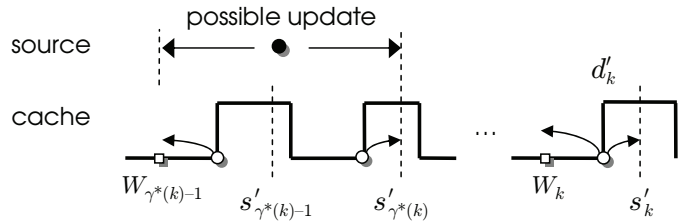


Fig. 11. Bounding the update age in ACT_1.

that sample points s_k that fall into OFF periods, as well as duplicate hits on the same ON cycle, provide no useful information about the update age A_U during preceding (or following) downloads from the source. Define s'_k to be the first sample point in the kth ON period observed by the probes and let d'_k be the preceding download timestamp at the cache. Then, d'_k is lower-bounded by

$$W_k = \max\left(s'_{k-1} + R_T\left(s'_{k-1}\right), \max\left(s_i : s_i < s'_k\right)\right), \tag{45}$$

which is the end of the $(k-1)$-st ON period or the preceding sample point, whichever happened later. Both cases are shown in Figure 10, where circles correspond to downloads and squares to their lower bounds.

Similar to passive sampling, suppose Δ'_k is a binary process that is 1 if a modification is detected during interval $(s'_{k-1}, s'_k]$, i.e., the record is different at download s'_k. We further define $\gamma^*(k) = \max(i \le k : \Delta'_i = 1)$ to be last sample (no later than k) that detected an update. Note that variables Δ_k and $\gamma(k)$ from passive measurement are unavailable in the active case, which is why they are replaced with those that can be computed by the observer, i.e., Δ'_k and $\gamma^*(k)$.

Now consider Figure 11. Note that the hidden download point d'_k always belongs to $[W_k, s'_k]$. This range is shown in the figure with a double arrow originating from d'_k. Moving $d_{\gamma^*(k)-1}$ as far back as possible and $d_{\gamma^*(k)}$ as far forward as possible (also shown with arrows) demonstrates that the last update must be contained in $[W_{\gamma^*(k)-1}, s'_{\gamma^*(k)}]$. Therefore, the age $A_U(d'_k)$ is bounded as

$$\max\left(W_k - s'_{\gamma^*(k)}, 0\right) \le A_U\left(d'_k\right) \le s'_k - W_{\gamma^*(k)-1}. \tag{46}$$

Note that the max function is needed when $\gamma'(k) = k$ to prevent the lower bound from going negative.

5.5 ACT_2

While (46) is a good low-overhead estimator, we can do even better with additional computation. Specifically, the main idea is to allow the estimator to utilize residuals $R_T(s'_k)$ *together* with the

already-recovered $G_T(x)$ to probabilistically determine the location of unknown download points d'_k. Recalling that $A_T(s'_k)$ is the age of the ON period at s'_k, it follows that $d'_k = s'_k - A_T(s'_k)$. The remaining elements of this approach, which we call ACT_2, is to determine the conditional distribution of $A_T(s'_k)$ and change the EM algorithm to work with random bounds $[L_k, R_k]$.

There are two pieces of information known to the observer that affect the distribution of $A_T(s'_k)$. The first is residual $y = R_T(s'_k)$ returned by the cache and the second is the upper bound $z = s'_k - W_k$ from preceding samples. Conditioning on $A_T < z$ and $R_T = y$, the tail distribution of A_T is

$$\bar{F}_A(x; y, z) := P(A_T > x | R_T = y, A_T < z) = \frac{P(x + y < T < y + z)}{P(y < T < y + z)}$$
$$= \frac{F_T(y + z) - F_T(x + y)}{F_T(y + z) - F_T(y)}, \tag{47}$$

where $F_T(x)$ comes from (43). Unless T is memoryless (i.e., exponential), parameter y provides useful clues about the possible values of age. For light-tailed distributions (e.g., constant, uniform), the age is generally a decreasing function of y. For heavy-tailed cases (e.g., Pareto), it is the opposite.

Leveraging (24), ACT_2 constrains the update age using *random* upper/lower bounds

$$L_k = s'_k - A_T(s'_k) - s'_{\gamma'(k)} + A_T(s'_{\gamma'(k)}) \tag{48}$$

$$R_k = s'_k - A_T(s'_k) - s'_{\gamma'(k)-1} + A_T(s'_{\gamma'(k)-1}), \tag{49}$$

where $A_T(s'_k) \sim F_A(x; R_T(s'_k), s'_k - W_k)$. Note that $\gamma'(k) = k$ implies that $L_k = 0$, which leaves only two random variables on the right side of (48)–(49). Otherwise, there are three of them, all with known parameters needed to construct (47).

For the computation, we discretize interval $[0, s'_k - W_k]$ and replace (47) with a PMF that assigns weights to a number of bins in that range. We then iterate over all possible ways to draw the three (or possibly two) age variables from their respective distributions and compute the probability ν_k for each deterministic bound $[l_k, r_k]$. These are fed into concave EM, which we modify to take weights into account, i.e., use

$$W_j(t) = \sum_{k=1}^{n} \frac{a_{jk}}{V_k(t)} \nu_k. \tag{50}$$

as a replacement for (42).

5.6 Overhead

The final issue is the cost of each algorithm. Define τ_n to be the amount of time needed to collect n samples of age A_U. In the passive case and $n \to \infty$, it is straightforward to infer that $\tau_n/n \to E[D] = E[T + Q]$. The active case requires more analysis.

Theorem 5.1. *Under active estimation of freshness f,*

$$\lim_{n\to\infty} \frac{\tau_n}{n} = \frac{E[T + Q]E[S]}{\int_0^\infty \bar{F}_S(x)\bar{F}_T(x)dx}, \tag{51}$$

where $\bar{F}_S(x)$ and $\bar{F}_T(x)$ are the complementary CDFs of inter-sample delays and TTLs, respectively.

Proof. Suppose active sampling generates M probe points $\{s_1, \ldots, s_M\}$ that produce n measurements of update age, which happens only at times that fall into ON periods and are the first ones therein. This happens if two conditions are present—(1) there is a cache hit at s_k; and (2) $A_T(s_k) < S_{k-1}$. Therefore, M has a negative binomial distribution and

$$\lim_{n\to\infty} \frac{n}{M} = hP(A_T < S). \tag{52}$$

Since $\tau_n = M/E[S]$, we have

$$\lim_{n\to\infty} \frac{\tau_n}{n} = \frac{E[S]}{hP(A_T < S)}, \tag{53}$$

which leads to (51) after expansion of the two terms in the denominator. □

Considering the result in (51), active sampling takes a constant factor longer than passive, where this ratio is

$$\frac{E[S]}{\int_0^\infty \bar{F}_S(x)\bar{F}_T(x)dx} \geq 1. \tag{54}$$

To visualize this better, notice that the inverse of (54) is

$$\frac{1}{E[S]} \int_0^\infty \bar{F}_S(x)\bar{F}_T(x)dx = P(R_S < T), \tag{55}$$

which specifies the probability that the residual inter-sample delay is smaller than the TTL. To enable quicker collection of useful measurements and achieve observation windows closer to those in passive sampling, $\text{ACT}_1/\text{ACT}_2$ require stochastically smaller R_S or stochastically larger T in order to bring (55) closer to 1. Interestingly, N_Q has no impact on overhead.

5.7 Other Metrics of Staleness

Besides freshness f, our measurement framework can be used to provide additional performance characterization of the system; in particular, that related to ***age of information*** (**AoI**) [32]. While none of the previous literature in AoI addresses the interval-censored estimation problems of this article, it might be useful to consider other definitions of age. One example is the average amount of time since the last update, i.e., source age $E[A_U(t)]$ at time t. This information is available to both passive and active observers once distribution $G_U(x)$ is estimated. Another way to leverage AoI is to consider the delay between the last download by the cache and the delivery of the object to the user, i.e., how long the object has been cached for. This is $E[A_T(t)]$, which is also easy to recover once $G_T(x)$ is known. Finally, neither of these definitions takes into account whether the object being served in stale or not. Thus, a more balanced metric might be *staleness lag* $L_1(t)$ from [34]. If the object is fresh at t, its staleness lag is zero; otherwise, it equals the amount of time the object has been stale for. Instead of evaluating the lag directly, it often makes sense to define a weight function $w(x)$ and consider *source penalty* [34]

$$\eta = E[w(L_1(t))]\mathbf{1}_{L_1(t)>0}. \tag{56}$$

This is a generalization of the framework considered so far, where $w(x) = 1$ produces the staleness probability $\eta = 1 - f$ and $w(x) = x$ yields staleness age $\eta = E[L_1(t)]$. With additional mild assumptions on N_U, [34, Theorem 6] shows that η is a convolution of the inter-update delay distribution $F_U(x)$ and inter-download delay distribution $F_D(x)$. In turn, since $D = T + Q$, we can obtain $F_D(x)$ through another convolution of $F_T(x)$ and Poisson $F_Q(x)$, where rate λ of $F_Q(x)$ is measured as part of the workflow in Figure 9.

6 Experiments

6.1 Setup

To investigate the accuracy of the developed sampling techniques, we registered an Internet domain and developed a custom authoritative server $\mathcal{A}$, written in C++, that could answer iterative IPv4 queries from arbitrary Internet hosts. Each DNS record (i.e., a hostname in our domain) that participated in the experiment was equipped with an update process N_U, which

changed the returned IP at points $\{u_i\}$. For passive monitoring, we created another C++ solution that ran a local DNS server $\mathcal{L}$, which accepted recursive/iterative queries from IPs in our subnet and resolved them at the authoritative server. Note that $\mathcal{A}$ and $\mathcal{L}$ were placed on different hosts.

For active sampling, we needed a remote server $\mathcal{R}$ that allowed recursive queries, returned non-fake answers, and complied with the source-provided TTL. To discover such options, we performed a port-53 UDP scan of the Internet (over 2.8B probed IPs) and found 8M responding hosts. Out of these, 3.7M (47%) reported no error and 3.2M (41%) supplied correct answers from $\mathcal{A}$. We then selected a subset of IPs from the last list, probed them for several days to ensure longevity, tested for TTL compliance, and screened against load-balancers that did not have a common cache. Out of the surviving options, we picked one at random to be $\mathcal{R}$.

Finally, we added into the mix an observer process and a background-traffic generator whose purpose was to query $\mathcal{L}$ or $\mathcal{R}$ depending on the scenario. The observer asked iterative queries after random delays $\{S_k\}$, while the generator sent packets from the query process N_Q. With the exception of network RTTs and various OS scheduling delays, the system functioned close to that in Figure 2 and allowed controlled experimentation with a known ground-truth.

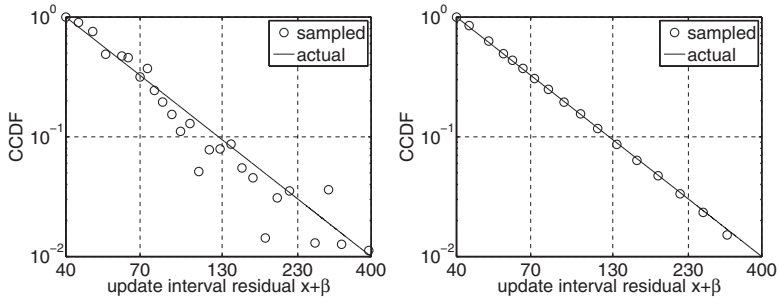
Unless mentioned otherwise, we kept the average delay between updates $E[U] = 20$ sec. It should be noted that $E[U]$ has little impact on the ability of the estimator to recover $F_U(x)$; instead, the predominant factor that matters is ability to collect observations of update age $A_U(t)$ from m unique update intervals, where m is sufficiently large. Informally speaking, as $E[U]$ becomes longer, the main downside is that it may take longer to observe the same m updates. Considering technical nuances of DNS, it can be further clarified that larger values of $E[U]$ produce better results since integer round-off errors in TTL are smaller in comparison to update cycle lengths. Thus, our experiments should be viewed as focusing on some of the more challenging scenarios, i.e., with a highly modified source (such as in Akamai CDNs, where 20-sec TTLs are common), and demonstrating that even in those cases the estimator works quite well. Also note that a similar observation applies to $E[S]$, $E[T]$, and $E[Q]$, i.e., these parameters impact the measurement window, but not the accuracy of the estimator.

The values of T dispatched from the authoritative server were uniform in $[1, 19]$ seconds, which mimicked Akamai-style churn rates and TTLs. Background clients sent traffic to the cache using a Poisson N_Q with rate $\lambda = 1$ query/sec and the active observer utilized exponential S with mean 4 sec. When a variable X needed to be Pareto-distributed, we drew it from $F(x) = 1 - (1 + x/\beta)^{-\alpha}$, where $x \geq 0$. We kept $\alpha = 3$ and $\beta = 2E[X]$ throughout all experiments, where $E[X]$ was the desired mean of the variable.

6.2 Passive Sampling

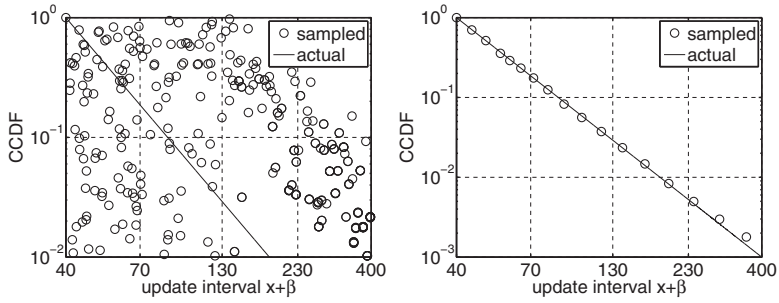
We start by examining recovery of $G_U(x)$ using our implementation of the local resolver $\mathcal{L}$. As shown in Figure 12(a) for Pareto U, method M_6 from [35] correctly identifies the trend of the tail, but the produced estimate is rather noisy. This approach is not well suited for such small samples sizes n. Applying the non-concave PASS from (27) yields a much better result in Figure 12(b). Its main drawback, however, is that conversion of the residual CDF into $F_U(x)$ is often impossible in practice. This is illustrated by the mishmash of points in Figure 13(a). Upgrading to the concave PASS from (30) leads to an amazingly better outcome in Figure 13(b).

Table 1 compares the runtime of M_6, the naive implementation of EM that directly computes (25)–(27), and our version of PASS in Algorithm 1. Observe that quadratic scaling of M_6 quickly makes it infeasible. In fact, in the last row of the table it requires an extrapolated 79 years to finish. The naive EM scales much better, although it still does not offer an appealing framework above 100K samples. On the other hand, PASS in the last column delivers blazingly fast results for all



(a) M_6 (b) non-concave EM ($\epsilon = 10^{-5}$)

Fig. 12. Passive estimation of $G_U(x)$ (n = 10K samples).



(a) non-concave EM ($\epsilon = 10^{-5}$) (b) concave EM ($\epsilon = 10^{-5}$)

Fig. 13. Passive estimation of $F_U(x)$ in PASS (n = 10K samples).

Table 1. Runtime in Passive Estimation of $G_U(x)$

n	M_6	Naive EM $\epsilon = 10^{-4}$	Algorithm 1 $\epsilon = 10^{-4}$
10^4	0.3 sec	9.4 sec	0.06 sec
10^5	58 sec	2.9 min	0.13 sec
10^6	2.2 hours	43 min	0.19 sec
10^7	–	5.5 hours	0.70 sec
10^8	–	–	5.79 sec
10^9	–	–	27.9 sec

input size up to 1B. While collecting this many observations in passive sampling is not likely in practice, recall that ACT_2 generates a huge number of deterministic bounds $[l_k, r_k]$ from (48)–(49). If age $A_T(s'_k)$ is discretized into 50 bins, a workload with 10K random bounds $[L_k, R_k]$ produces 500M intervals for concave EM. Therefore, Algorithm 1 is by far the only feasible way to compute the ACT_2 estimator.

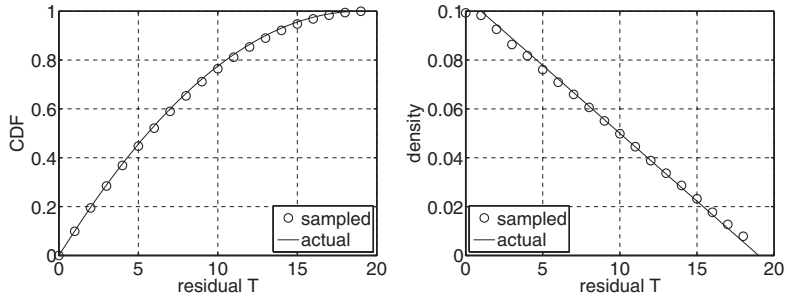
Another interesting question is the number of iterations $t_\epsilon(n)$ needed for PASS to converge as a function of n and ϵ, which is shown in Table 2 together with the runtime. Predictably, smaller ϵ causes more iterations, but the execution delay does not always increase in proportion to $t_\epsilon(n)$. For example, the last column has $t_\epsilon(n)$ increasing by 200 times; however, the runtime goes up by only 39%. This can be explained by the small number of iterations, where the CPU cost is still

Table 2. Effect of n and ϵ on PASS (Runtime in Sec)

ϵ	$n = 10^3$		$n = 10^5$		$n = 10^7$	
	$t_\epsilon(n)$	runtime	$t_\epsilon(n)$	runtime	$t_\epsilon(n)$	runtime
10^{-2}	38	0.0001	5	0.0001	4	0.531
10^{-3}	216	0.015	119	0.016	56	0.546
10^{-4}	562	0.032	538	0.047	289	0.608
10^{-5}	1,488	0.063	1,360	0.124	800	0.749

Table 3. Relative Estimation Error of Simple Parameters

n	ACT_1 & ACT_2			PASS		
	h	$E[T]$	λ	h	$E[T]$	λ
10^2	2.94%	16.8%	51.6%	1.58%	4.18%	7.93%
10^3	0.88%	5.81%	11.5%	0.42%	1.58%	2.79%
10^4	0.29%	2.40%	3.73%	0.16%	0.41%	0.80%



(a) $G_T(x)$ (b) $g_T(x)$

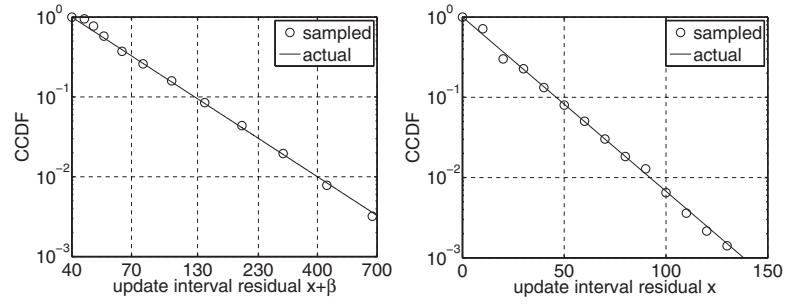
Fig. 14. Distribution of TTL in active sampling (n = 10K samples).

dominated by the initial sort. Additionally, notice that larger n allows for faster convergence since the data is less noisy and the solution is found quicker.

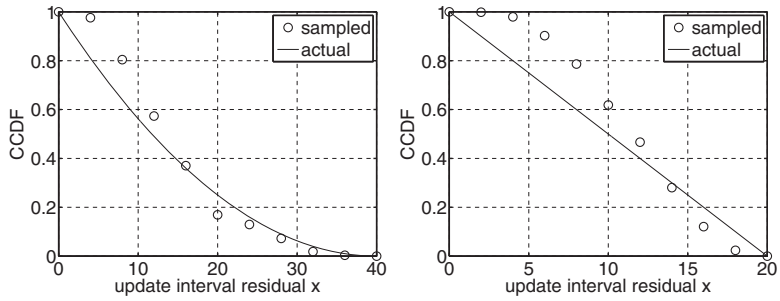
6.3 Active Sampling

We first analyze accuracy of recovery for $G_T(x)$, h, $E[T]$, and λ from Figure 9 for which $\text{ACT}_1/\text{ACT}_2$ have a common estimation algorithm. The result for $G_T(x)$ and its density $g_T(x)$ is given by Figure 14, which indicates a strong match. Note that usage of concave EM is a must for sampling the TTL distribution since $E[T] = 1/g_T(0)$ is needed for computation of λ. Table 3 displays the remaining shared parameters and also compares against PASS. Note that the latter method measures these quantities locally, while the other two sample them remotely. As expected, PASS's averages converge quicker, although active measurement still produces solid results. Another interesting fact is that λ in the fourth column is highly sensitive to errors in h, which comes from its shape $\lambda = h/(1-h)/E[T]$.

We next evaluate estimation accuracy of $G_U(x)$. Figure 15 shows the output of ACT_1. While the method appears to handle Pareto and exponential U rather well, there is non-negligible discrepancy for small x. As the tail of $G_U(x)$ becomes lighter in subfigures (c)-(d), this deviation is easier to see. On the other hand, ACT_2 in Figure 16 recovers these distributions with significantly better accuracy. A further confirmation of these findings is given by Tables 4–7, where ACT_1 not

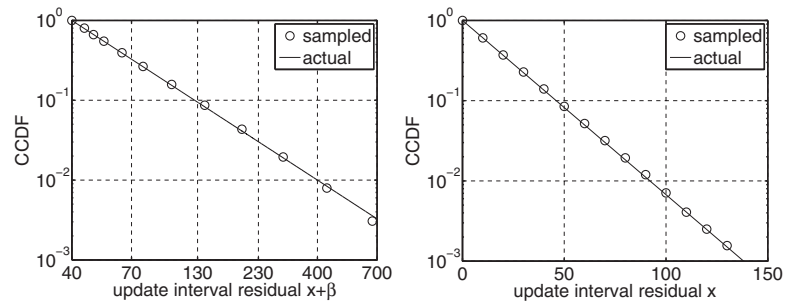


(a) Pareto U (b) exponential U

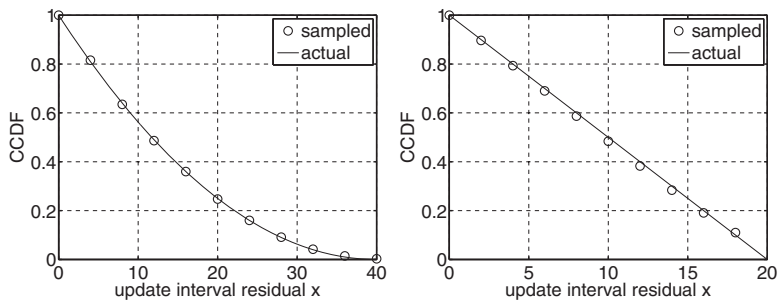


(c) uniform U (d) constant U

Fig. 15. Tail of $G_U(x)$ in ACT_1 (n = 10K samples).



(a) Pareto U (b) exponential U



(c) uniform U (d) constant U

Fig. 16. Tail of $G_U(x)$ in ACT_2 (n = 10K samples).

Table 4. Relative Error under Pareto U

n	ACT_1		ACT_2		PASS	
	p	f	p	f	p	f
10^2	14.4%	12.8%	6.5%	5.8%	6.3%	5.6%
10^3	12.6%	11.2%	1.9%	1.8%	1.9%	1.7%
10^4	12.2%	10.8%	0.5%	0.4%	0.7%	0.6%

Table 5. Relative Error under Exponential U

n	ACT_1		ACT_2		PASS	
	p	f	p	f	p	f
10^2	16.2%	14.4%	5.6%	5.3%	5.0%	4.4%
10^3	14.3%	12.6%	1.8%	1.6%	1.7%	1.5%
10^4	13.9%	12.3%	0.6%	0.5%	0.6%	0.5%

Table 6. Relative Error under Uniform U

n	ACT_1		ACT_2		PASS	
	p	f	p	f	p	f
10^2	18.3%	16.0%	5.0%	4.8%	4.0%	3.4%
10^3	16.1%	14.2%	1.7%	1.5%	1.2%	1.1%
10^4	15.8%	13.9%	0.5%	0.4%	0.5%	0.5%

Table 7. Relative Error under Constant U

n	ACT_1		ACT_2		PASS	
	p	f	p	f	p	f
10^2	19.2%	17.0%	5.8%	5.2%	3.8%	3.4%
10^3	16.9%	14.8%	1.8%	1.6%	1.1%	1.0%
10^4	16.8%	14.7%	1.1%	0.8%	0.4%	0.4%

Table 8. Measurement-Delay Error under Pareto U

n	ACT_1 & ACT_2	PASS
10^2	3.68%	3.92%
10^3	1.12%	1.24%
10^4	0.35%	0.39%
10^5	0.11%	0.12%
10^6	0.04%	0.04%

only suffers from poor estimation accuracy, but also fails to improve with larger n. This is caused by a non-diminishing bias in $G_U(x)$.

Analysis of error for the measurement delay τ_n/n is shown in Table 8, where the passive case uses $E[T + Q]$ as the convergence point of τ_n/n and active employs the result of Theorem 5.1. Note that PASS requires on average 11 sec per usable sample, while the other two methods spends 21% more (i.e., 13.3 sec). An experiment with $n = 10$K samples consumes an expected 30.5 hours in the former case and 36.9 hours in the latter. The active-probing bandwidth (i.e., 3.3 packets per

sample of A_U) is negligible and does not contribute to the runtime. As further shown in the table, the derived cost models are quite accurate, even for small n.

Overall, concave PASS emerges as hands-down the best tool for sampling update dynamics and staleness in *single-blind* scenarios (i.e., only the update process is hidden) and ACT_2 does the same in *double-blind* (i.e., both update/download processes are invisible to the observer).

7 Conclusion

We presented a general framework for modeling freshness in TTL-based caching systems, proposed three novel techniques for remotely measuring this metric in the current Internet, and improved the performance of existing update-sampling algorithms. Even under random TTL, our most advanced method can recover all unknown parameters of the system (i.e., hit and request rate, update/TTL distributions, and freshness), without requiring any change to the DNS infrastructure.

References

[1] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman. 2012. A survey of information-centric networking. *IEEE Communications Magazine* 50, 7 (Jul. 2012), 26–36.

[2] Hüseyin Akcan, Torsten Suel, and Hervé Brönnimann. 2008. Geographic web usage estimation by monitoring DNS caches. In *LocWeb*.

[3] Sadiye Alici, Ismail Sengor Altingovde, Rifat Ozcan, B. Barla Cambazoglu, and Özgür Ulusoy. 2012. Adaptive time-to-live strategies for query result caching in web search engines. In *ECIR*.

[4] Hussein A. Alzoubi, Michael Rabinovich, and Oliver Spatscheck. 2013. The anatomy of LDNS clusters: Findings and implications for web content delivery. In *WWW*. 83–94.

[5] Omri Bahat and Armand M. Makowski. 2005. Measuring consistency in TTL-based caches. *Performance Evaluation* 62, 1–4 (Oct. 2005), 439–455.

[6] Daniel S. Berger, Philipp Gland, Sahil Singla, and Florin Ciucu. 2014. Exact analysis of TTL cache networks. *Performance Evaluation* 79 (Apr. 2014), 2–23.

[7] Daniel S. Berger, Ramesh K. Sitaraman, and Mor Harchol-Balter. 2017. AdaptSize: Orchestrating the hot object memory cache in a content delivery network.. In *NSDI*. 483–498.

[8] Roi Blanco, Edward Bortnikov, Flavio Junqueira, Ronny Lempel, Luca Telloli, and Hugo Zaragoza. 2010. Caching search engine results over incremental indices. In *ACM SIGIR*. 82–89.

[9] Mark Brown. 1980. Bounds, inequalities, and monotonicity properties for some specialized renewal processes. *The Annals of Probability* 8, 2 (1980), 227–240.

[10] Thomas Callahan, Mark Allman, and Michael Rabinovich. 2013. On modern DNS behavior and properties. *ACM CCR* 43, 3 (Jul. 2013), 7–15.

[11] Berkant Barla Cambazoglu, Flavio P. Junqueira, Vassilis Plachouras, Scott Banachowski, Baoqiu Cui, Swee Lim, and Bill Bridge. 2010. A refreshing perspective of search engine caching. In *WWW*. 181–190.

[12] Valeria Cardellini, Michele Colajanni, and S. Yu Philip. 1999. DNS dispatching algorithms with state estimators for scalable Web-server clusters. *World Wide Web* 2, 3 (1999), 101–113.

[13] Valeria Cardellini, Michele Colajanni, and Philip S. Yu. 1999. Dynamic load balancing on web-server systems. *IEEE Internet Computing* 3, 3 (1999), 28–39.

[14] Devarshi Chatterjee, Zahir Tari, and Albert Zomaya. 2005. A task-based adaptive TTL approach for web server load balancing. In *ISCC*. 877–884.

[15] Chen Chen, Stephanos Matsumoto, and Adrian Perrig. 2015. ECO-DNS: Expected consistency optimization for DNS. In *IEEE ICDCS*. 256–267.

[16] Xin Chen, Haining Wang, Shansi Ren, and Xiaodong Zhang. 2006. DNScup: Strong cache consistency protocol for DNS. In *IEEE ICDCS*.

[17] K. L. Chung. 1974. *A Course in Probability Theory* (2nd ed.). Academic Press.

[18] Edith Cohen and Haim Kaplan. 2001. The age penalty and its effect on cache performance. In *USENIX USITS*. 73–84.

[19] Edith Cohen and Haim Kaplan. 2001. Aging through cascaded caches: Performance issues in the distribution of web content. In *ACM SIGCOMM*. 41–53.

[20] Edith Cohen and Haim Kaplan. 2001. Refreshment policies for web content caches. In *IEEE INFOCOM*. 1398–1406.

[21] Edith Cohen and Haim Kaplan. 2003. Proactive caching of DNS records: Addressing a performance bottleneck. *Computer Networks* 41, 6 (Oct. 2003), 707–726.

[22] Michele Colajanni and Philip S. Yu. 2002. A performance study of robust load sharing strategies for distributed heterogeneous web server systems. *IEEE Transactions on Knowledge and Data Engineering* 14, 2 (2002), 398–414.
[23] Yuguang Fang, Zygmunt J. Haas, Ben Liang, and Yi-Bing Lin. 2004. TTL prediction schemes and the effects of inter-update time distribution on wireless data access. *Wireless Networks* 10, 5 (Sep. 2004), 607–619.
[24] Kassem Fawaz and Hassan Artail. 2013. DCIM: Distributed cache invalidation method for maintaining cache consistency in wireless mobile networks. *IEEE Transactions on Mobile Computing* 12, 4 (Apr. 2013), 680–693.
[25] Nicaise Choungmo Fofack and Sara Alouf. 2013. Modeling modern DNS caches. In *ValueTools*.
[26] Nicaise Choungmo Fofack, Philippe Nain, Giovanni Neglia, and Don Towsley. 2014. Performance evaluation of hierarchical TTL-based cache networks. *Computer Networks* 65 (Jun. 2014), 212–231.
[27] Christine Fricker, Philippe Robert, and James Roberts. 2012. A versatile and accurate approximation for LRU cache performance. In *ITC*. 8:1–8:8.
[28] Hazem Gomaa, Geoffrey G. Messier, and Robert Davies. 2015. Hierarchical cache performance analysis under TTL-based consistency. *IEEE/ACM Transactions on Networking* 23, 4 (Aug. 2015), 1190–1201.
[29] Y. Thomas Hou, Jianping Pan, Bo Li, and Shivendra S. Panwar. 2004. On expiration-based hierarchical caching systems. *IEEE JSAC* 22, 1 (Jan. 2004), 134–150.
[30] Collin Jackson, Adam Barth, Andrew Bortz, Weidong Shao, and Dan Boneh. 2007. Protecting browsers from DNS rebinding attacks. In *ACM CCS*.
[31] Jaeyeon Jung, Arthur W. Berger, and Hari Balakrishnan. 2003. Modeling TTL-based internet caches. In *IEEE INFOCOM*. 417–426.
[32] S. Kaul, M. Gruteser, V. Rai, and J. Kenney. 2011. Minimizing age of information in vehicular networks. In *IEEE SECON*. 350–358.
[33] Jeong-Joon Lee, Kyu-Young Whang, Byung Suk Lee, and Ji-Woong Chang. 2002. An update-risk based approach to TTL estimation in web caching. In *IEEE WISE*. 21–29.
[34] X. Li, D. B. H. Cline, and D. Loguinov. 2016. On sample-path staleness in lazy data replication. *IEEE/ACM Transactions on Networking* 24, 5 (Oct. 2016), 2858–2871.
[35] Xiaoyong Li, Daren B. H. Cline, and Dmitri Loguinov. 2017. Temporal update dynamics under blind sampling. *IEEE/ACM Transactions on Networking* 25, 1 (Feb. 2017).
[36] Guoxin Liu, Haiying Shen, Harrison Chandler, and Jin Li. 2014. Measuring and evaluating live content consistency in a large-scale CDN. In *IEEE ICDCS*. 268–277.
[37] Xiaobo Ma, Junjie Zhang, Zhenhua Li, Jianfeng Li, Jing Tao, Xiaohong Guan, John C. S. Lui, and Don Towsley. 2015. Accurate DNS query characteristics estimation via active probing. *Journal of Network and Computer Applications* 47 (Jan. 2015), 72–84.
[38] Valentina Martina, Michele Garetto, and Emilio Leonardi. 2014. A unified approach to the performance analysis of caching systems. In *IEEE INFOCOM*. 2040–2048.
[39] Jarmo Mölsä. 2004. Mitigating DoS attacks against the DNS with dynamic TTL values. In *NordSec*. 118–124.
[40] Chris Olston and Jennifer Widom. 2002. Best-effort cache synchronization with source cooperation. In *ACM SIGMOD*. 73–84.
[41] Jeffrey Pang, Aditya Akella, Anees Shaikh, Balachander Krishnamurthy, and Srinivasan Seshan. 2004. On the responsiveness of DNS-based network control. In *ACM IMC*. 21–26.
[42] Moheeb Abu Rajab, Fabian Monrose, Andreas Terzis, and Niels Provos. 2008. Peeking through the cloud: DNS-based estimation and its applications. In *ACNS*.
[43] S. Resnick. 1999. *A Probability Path*. Birkhäuser.
[44] Fethi Burak Sazoglu, B. Barla Cambazoglu, Rifat Ozcan, Ismail Sengor Altingovde, and Özgür Ulusoy. 2013. Strategies for setting time-to-live values in result caches. In *ACM CIKM*. 1881–1884.
[45] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. 2013. On measuring the client-side DNS infrastructure. In *ACM IMC*. 77–90.
[46] Haiying Shen. 2010. IRM: Integrated file replication and consistency maintenance in P2P systems. *IEEE Transactions on Parallel and Distributed Systems* 21, 1 (Jan. 2010), 100–113.
[47] Kristinn Sigurosson. 2005. Incremental crawling with Heritrix. In *IWAW*.
[48] Yiu Fai Sit, Francis C. M. Lau, and Cho-Li Wang. 2005. On the cooperation of web clients and proxy caches. In *IEEE ICPADS*. 264–270.
[49] Raghav Srinivasan, Chao Liang, and Krithi Ramamritham. 1998. Maintaining temporal coherency of virtual data warehouses. In *IEEE RTSS*.
[50] Xueyan Tang, Jianliang Xu, and Wang-Chien Lee. 2008. Analysis of TTL-based consistency in unstructured peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems* 19, 12 (Dec. 2008), 1683–1694.
[51] Bruce W. Turnbull. 1976. The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society* 38, 3 (1976), 290–295.

[52] Craig E. Wills, Mikhail Mikhailov, and Hao Shang. 2003. Inferring relative popularity of internet applications by actively querying DNS caches. In *ACM IMC*. 78–90.
[53] Joel L. Wolf, Mark S. Squillante, Philip S. Yu, Jay Sethuraman, and L. Ozsen. 2002. Optimal crawling strategies for web search engines. In *WWW*. 136–147.
[54] R. W. Wolff. 1989. *Stochastic Modeling and the Theory of Queues*. Prentice Hall.
[55] D. Xiao, X. Li, D. B. H. Cline, and D. Loguinov. 2018. Estimation of DNS source and cache dynamics under interval-censored age sampling. In *IEEE INFOCOM*. 1358–1366.
[56] Yaming Yu. 2011. Concave renewal functions do not imply DFR inter-renewal times. *Journal of Applied Probability* 48, 2 (Jun. 2011), 583–588.

Received 31 January 2024; revised 11 November 2024; accepted 8 January 2025