Performance Aspects Of Distributed Caches Using TTL-based Consistency *

Edith Cohen
AT&T Labs–Research
180 Park Avenue, Florham Park, NJ 07932
edith@research.att.com

Eran Halperin   Haim Kaplan
Tel Aviv University
Tel Aviv 69978, Israel
{heran,haimk}@math.tau.ac.il

September 16, 2003

Abstract

The web is the largest distributed database deploying TTL-based weak consistency. Each object has a lifetime-duration assigned to it by its origin server. A copy of the object fetched from its origin server is received with maximum time-to-live (TTL) that equals its lifetime duration. In contrast a copy obtained through a cache have shorter TTL since the *age* (elapsed time since fetched from the origin) is deducted from its lifetime duration. A request served by a cache constitutes a *hit* if the cache has a fresh copy of the object. Otherwise, the request is considered a *miss* and is propagated to another server. It is evident that the number of cache misses depends on the age of the copies the cache receives. Thus, a cache that sends requests to another cache would suffer more misses than a cache that sends requests directly to an authoritative server.

In this paper we model and analyze the effect of age on the performance of various cache configurations. We consider a low-level cache that fetches objects either from their origin servers or from other caches and analyze its miss rate as function of its fetching policy. We distinguish between three basic fetching policies, namely, fetching always from the origin, fetching always from the same high-level cache, and fetching from a "random" high-level cache. We explore the relationships between these policies in terms of the miss rate achieved by the low-level cache, both on worst-case sequences, and on sequences generated using particular probability distributions.

Guided by web caching practice, we consider two variations of the basic policies. In the first variation the high-level cache uses *pre-term refreshes* to keep a copy with lower age. In the second variation the low-level cache uses *extended lifetime duration*. We analyze how these variations affect the miss rates. Our theoretical results help to understand how age may affect the miss rate, and imply guidelines for improving performance of web caches.

Keywords: caching hierarchy, cache consistency, Time To Live (TTL), Web caching, miss rate.

*Some of the results reported here first appeared in the proceedings of ICALP'01 [2].

1 Introduction

The web is the largest example of a distribution system deploying TTL-based consistency. web objects are typically associated with one authority that originates and modifies them (their *authoritative* server), but can be cached and further distributed from multiple *replicating* servers (caches). Caching and replication are widely deployed for reducing load on web servers, network load, and user-perceived latency. Caches are located at different points in the network and include reverse proxies, proxy caches, and browser caches. By and large, servers rely on one widely supported and deployed mechanism for consistency of cached copies. This client-driven and expiration-based mechanism is specified in HTTP (Hyper-Text Transfer Protocol) [5] which governs the distribution of web content: The authoritative server provides an expiration time for each copy, beyond which it must be validated or discarded.

Since the web application is the underlying motivation for our work, we further elaborate on HTTP consistency. Each object has a URL which specifies its "location" and its authoritative server. The object is requested by sending an HTTP request and the content is sent back on the respective HTTP response. The response includes a header with important information on the object, including cache directives. The directives specify if the object can be cached, and may provide explicit expiration time or information that can be used to compute one. When an object is requested from the cache then if the cache has a *fresh* (non-expired) copy, the request is processed locally. If the cached copy is *stale* (expired), it must be validated by contacting a server with a fresh copy. To this end, HTTP provides conditional `GET` requests. Similarly, if there is no cached copy, the cache must obtain a fresh copy. Requests for which the cache does not have a fresh copy and thus must contact an external server are termed *cache misses*.

An HTTP-compliant cache [1, 5, 14] calculates from the header fields of an object a *freshness lifetime*, which is the time interval during which the copy remains fresh (non-expired) since it left its authoritative server. The freshness lifetime is typically fixed for each object. The cache also determines from the headers the *age* of the copy, which is the elapsed time since it left its authoritative server (and resided in caches). If the age is smaller than the freshness lifetime then by subtracting the age of the copy from its freshness lifetime the cache obtains a *time-to-live (TTL)* duration for the copy (during which it is considered fresh). If the age is larger than the freshness lifetime, the object is considered stale and its TTL is zero.

Web caches are placed both as proxy caches close to clients and reverse proxies close to web servers [8, 14]. They are also sometimes configured in hierarchies [9]. Thus, often there is more than one cache placed between the end user and the origin server. Different copies corresponding to the same URL and residing in different caches can have different ages and thus different TTLs. In particular, a copy obtained by a cache from another cache expires sooner than a copy obtained from an authoritative server. Therefore, a cache that forwards requests to another cache is likely to suffer more cache misses than a cache that forwards requests to an authoritative server. Furthermore, a cache is likely to obtain more client requests than an authoritative server since it serves older copies. So even if a cache maintains fresh copies, since the copies are aged, the cache is somewhat less effective than an authoritative server [3, 4]. We identify age as a central performance factor where TTL-based consistency is used, as it affects frequently-requested objects, and it prevails even with infinite storage.

In this paper we model and analyze how the miss rate of a client cache depends on the source from which it obtains its objects. We explore three different basic policies. According to the first policy the client cache uses an authoritative server directly without a high-level cache. In the second configuration the client cache uses consistently for all its misses the same source cache. In the third configuration the client cache alternates between several source caches to resolve its misses. We model these policies and the TTL distribution through each of them, in a way that allows rigorous analytical analysis.

We show that a cache which consistently uses the same source cache (for each object) would incur a lower miss rate than a cache that alternates between several caches, on any sequence of requests. This suggests that when a set of caches handles the workload of a set of clients, it is better to partition the workload such that the same primary cache is used for each client-object pair.

We characterize precisely how worse the miss rate could be when using a source cache compared to using an authoritative server on the worst possible sequence. For a single consistent source we show that the miss rate is larger by at most a factor of 2. For alternating between sources, the miss rate could be somewhat worse, but not more than a factor of $e = 2.718..$ larger than the miss rate through an authoritative server.

We define extensions of the basic policies that incorporate two further cache behaviors. The basic models assumed that only expired copies can be refreshed. HTTP caches, however, may refresh fresh items.[1] Conceivably, a cache can deploy configured periodic pre-term refreshes (*rejuvenations*) to reduce the age of cached copies and as a result, improve the miss-rate at its client caches and consequently, the number of requests it receives. Rejuvenation increases traffic between a cache and the authoritative server but can decrease traffic between the cache and its clients. As

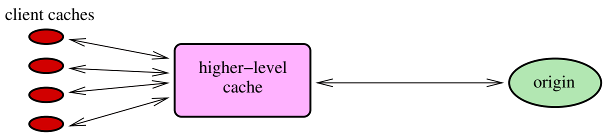

Figure 1: Client caches, higher-level cache, and origin server

illustrated in Figure 1, a single cache can potentially serve a large number of (caching) clients, and thus rejuvenation can be very effective. As pre-term refreshes decrease the average age of cached copies, it may seem that they can only increase the effectiveness of a cache. We show, however, that this is generally not the case. In particular we show that when pre-term refreshes occur in arbitrary points in time and "out of synch" with the expiration times of previous copies, performance can degrade by up to a factor of 2, which is tight under some robust conditions. We also show however, that when used "in synch" with previous expirations, pre-term refreshes can boost performance.

Although not typically the case, caches often need to apply a heuristic freshness lifetime calculation that results in different freshness lifetime durations for the same copy in different caches. A second extension of the basic model considers caches that use longer lifetime durations than their source. These caches can improve their miss-rate albeit at the cost of increased staleness. We show that for many prototypical request sequences, the tradeoff of staleness vs. miss-rate is not convex, and that specifically, integral values of the *extension factor* (the ratio of the lifetime time durations at the cache and its source) are more effective. As an analysis tool, we establish a correspondence between rejuvenating sources and extended lifetime caches, and use this correspondence to prove relationships between the various sources.

Finally, we consider objects with inter-request times drawn from natural known probability distributions, namely, fixed inter-request times and Poisson arrivals. We analyze the performance

[1]Such *pre-term refreshes* occur when a client request contains a *no-cache* request header. When a cache received such a request it must forward it to the origin server even if it has a fresh copy. The cache uses the response to replace or refresh its older copy of the object.

of a cache that receives these request sequences under the different configurations of higher-level caches and with and without lifetime extension.

Related work: Although age is a central performance factor where TTL based consistency is used through a hierarchy, we are not aware of any previous work where it is being modeled and analyzed analytically.

TTL-based consistency is used also in the Domain Name System [12, 13], where each resource record (RR) in the DNS database has a freshness lifetime duration assigned to it by its authoritative administrator. Recent work of Jung *et al* [11] explores the effect of the length of the freshness lifetime on the miss-rate. They show that relatively short TTLs suffice to get most of the hits. Some aspects of our study may be relevant to the DNS system as well. For example when there is a choice between using one or alternating between several authoritative servers for some zone.

Subsequent to the first appearance of this work [2], Jung *et al* [10] supported their experimental work on DNS traces [11] with an analytical analysis. In this work they model requests as independent identically distributed random variable drawn from some fixed probability distribution. They develop formula for the miss rate generated by several particular probability distributions and analyze how freshness lifetime duration affects this miss-rate. For the probability distributions that fit best real traces they obtained results similar to the ones observed from the traces alone. That is, relatively short TTLs suffice to get low miss rate and the marginal affect of increasing the TTL further is small.

Another recent work of Hou *et al* [6, 7] focus on cache hierarchies with more than two levels. For requests generated by a Poisson distribution they calculate the miss-rate and average TTL at different levels of the hierarchy. Their analysis quantifies how the TTL shortens, and the miss rate increase, as we go to deeper caches in the hierarchy. In their more recent work [7] they also analyze the effect of a rejuvenation mechanism, similar to ours, on the TTL at various levels.

In a complementary work [4] we demonstrate some of the analytical findings of this paper using different inter-request time distributions and simulations based on real traces.

The structure of the rest of the paper is as follows. Section 2 gives an overview of our caching model. Section 3 provides general results for the three basic models and establishes their relative performance. Section 4 analyzes pre-term refreshes and freshness-lifetime extensions. Section 5 includes worst-case analysis that provides tight bounds on the competitive ratio of different configurations. In Section 6 we analyze Poisson and fixed-frequency arrivals. We conclude in Section 7.

2 TTL-based Caching: Model and Policies

In this section we define the model which we use to analyze age related performance issues for different cache configurations. Recall that a cache upon a miss may fetch the object from the origin server or from another cache. We formally define three basic policies (which we call "sources") for a cache to fetch up an object upon a miss. In later sections we analyze how these policies and their extensions may affect the miss rate of the cache. Combinations of these policies can easily model real cache configurations.

Consider a set of distributed servers that originate, cache, request, and distribute copies of an object. The object can be modified over time but only by its origin server. Each copy of an object has a designated *lifetime* duration which after it elapses, the copy can not be further distributed or used. The *lifetime* value is assigned by the origin server when it serves the copy. Our analysis focuses on lifetime values that are fixed throughout time. That is, the origin always provides lifetime of T. The *age* of a cached copy is the elapsed time since it was obtained from an origin server. The TTL of a cached copy equals the lifetime T minus its age. If the age is larger than T, the copy is *stale*. Otherwise, the copy is *fresh*. We differentiate between *origin* (authoritative) servers and *caches* (replicating servers). The object can be requested from any server. Origin servers always provide a copy with zero age and thus a *time-to-live* (TTL) of T. Caches process the request by providing a local fresh copy, if there is one. The request then is considered as a *cache hit*. Otherwise, another server is contacted and the response can be used to update the cached copy. The request then is a *cache miss*. The *miss rate* of cache is the fraction of cache misses among the total number of requests.

We use the term *source* for the entity to which a client-cache sends requests. A source is defined by a set of one or more servers, the strategy they use to refresh their copies, and the pattern used by the client-cache to select a server. We are interested in how the type of the source affects the miss rate at the client-cache.

We assume objects always remain in the cache until they expire. In particular, there is no cache capacity constraint and no object can trigger an eviction of another, so the behavior of the cache on different objects is independent. Therefore, our analysis considers requests for a single object. Since we focus on age-induced effects, we generally consider sources that always provide a fresh, but possibly aged, copy of the object.

2.1 Basic Source Types

We define three types of *sources* that capture different relationships between a client-cache and its data source(s). The different sources are illustrated in Figure 2. The TTL value obtained through

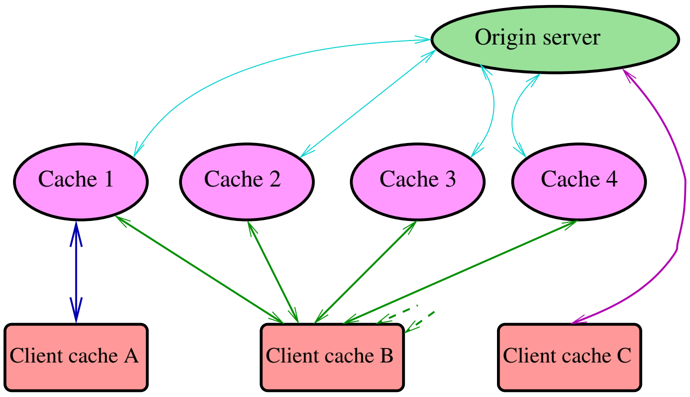

Figure 2: Different types of sources: Cache A uses cache 1 as an EXC source. Cache B uses caches 1,2,... as an IND source. Cache C uses an AUTH source.

each source as a function of time is illustrated in Figure 3.

1. AUTH: An authoritative source. AUTH always provides a copy with zero age (TTL that equals the freshness lifetime).

2. EXC: Let α be drawn uniformly at random from the interval $[0, T]$ (the distribution $U[0, T]$), where T is the lifetime of the object. We call α the *displacement*. At time t an EXC source provides a copy whose age is $(t - \alpha) \bmod{}^2 T$ (TTL equals to $T - (t - \alpha) \bmod T$). Note that

[2] "mod" is a generalized modulus operation to arbitrary nonnegative numbers $a \bmod b = a - b * \lfloor a/b \rfloor$. If $b = 0$ then we define $a \bmod b \equiv 0$.

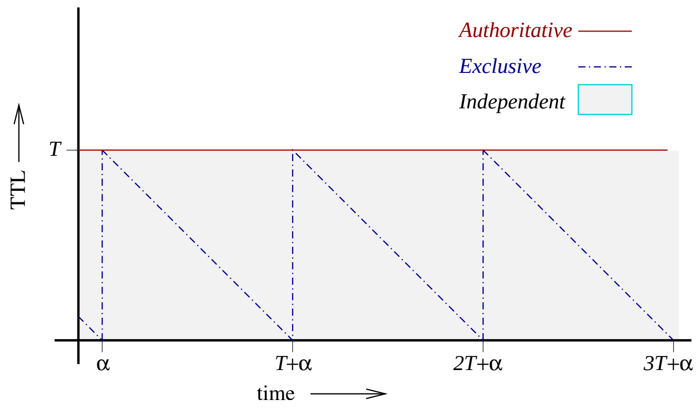

Figure 3: TTL obtained through different types of sources. AUTH source always provides zero age and TTL of T. EXC source (shown with displacement α) provides age that cycles from 0 to T and thus TTL which cycles from T to 0. IND source provides at each point in time age drawn independently from $U[0, T]$ and thus TTL drawn from $U[0, T]$.

the choice of α is fixed for each "run" on the sequence, and the performance of EXC is the *expected* performance over all runs with different displacements.

This source models a scenario where the client-cache upon each miss fetches a copy from the *same* high-level-cache. This high-level-cache maintains a fresh copy by refreshing it through an AUTH source each time it expires. The averaging over different displacements captures independence of the request times at the client-cache and the particular "refresh period" of the high-level-cache.

3. IND: Upon each miss at the client-cache, the IND source provides a copy with age independently drawn from $U[0, T]$ (thus a TTL that is also drawn from $U[0, T]$).

   This source models a scenario where upon each miss, the client-cache forwards the request to a different independent EXC-type high-level-cache. Independence means that the displacements of the different high-level-caches are not correlated.

The EXC, IND, and AUTH sources model pure scenarios. Currently, many web caches direct requests for a particular URL through a high-level-cache (e.g., a proxy or a reverse proxy) and hybrids of the IND and EXC sources capture scenarios where several caches are used for load balancing purposes.

Our analysis assumes that the request sequence is independent of the performance of the client-cache (that is, future requests do not depend on which previous requests constituted a hit or a miss).

2.2 Consistency and synchronization

In our analysis we use the notion of source consistency. A source is *consistent* if the age provided at any given time is consistent with past values. Formally, if at time t, the source provided an age of Φ then at time $t + \Delta$ the source provides the object with age of at most $\Phi + \Delta$. The AUTH and EXC sources are consistent. In general, we expect sources that model a single server to be consistent.

Synchronization is a property of the relationship between a cache and its source. We say that a client-cache is *synchronized* with a source if whenever the client-cache contains a copy of the object which expires at some time t, then requests directed to the source at times $t + \Delta$ ($\Delta > 0$) obtain an object whose age is not more than Δ. By definition, a client-cache is always synchronized with AUTH and EXC sources but not with an IND source. As we shall see, synchronization helps performance. Intuitively, synchronization means that the copy at the source expires at the same time as the copy at the client-cache, and thus, misses at the client-cache on requests which closely follow previous requests are more likely to yield a copy with small age.

2.3 Extended Lifetime at a Client Cache

We consider situations where a client cache uses a longer lifetime duration than used by the source. As elaborated in the introduction, longer lifetimes can be the outcome of different settings or simply an attempt by the client-cache to reduce its miss-rate and decrease traffic between itself and its source. Lifetime extension trades increased miss-rate with increased staleness. We consider client-

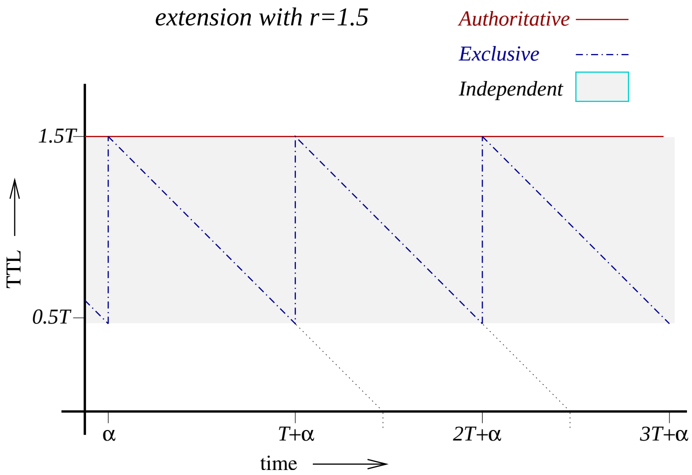

Figure 4: TTL obtained through different types of sources when the client cache uses extension factor of $r = 1.5$. AUTH source provides copies with zero age and thus TTL of rT ($1.5T$). EXC source (shown with displacement α) provides copies with age that cycles from 0 to T and thus a TTL that cycles from rT to $(r-1)T$ ($1.5T$ to $0.5T$). IND source provides at each point in time a copy with age independently drawn from $U[0, T]$ and thus TTL drawn from $U[(r-1)T, rT]$ ($U[0.5T, 1.5T]$). The dotted lines correspond to expiration times of previous copies at the client cache. Copies expire at times $(i + 0.5)T + \alpha$ for integral values of i.

caches that use an extended lifetime value of $r * T$ for some fixed $r > 1$ (As before, T is the lifetime value as used by the source). We refer to r as the *extension factor*. We use the notation AUTH(r) (respectively, IND(r), EXC(r)) for a source of type AUTH (respectively, IND, EXC) in conjunction with a client-cache applying an extension factor of r. The TTL value for the different source types and extension factor of 1.5 are illustrated in Figure 4. Note that a client-cache is synchronized with its EXC(r) source if and only if r is integral. Figure 4 illustrates the lack of synchronization with $r = 1.5$: when previously-served copies expire at the client cache, the source provides a copy with age $0.5T$ and thus a TTL of T.

2.4 Rejuvenating Sources

We consider replicating servers that refresh selected objects as soon as their TTL drops below some threshold (rather than wait for it to completely expire). We refer to such configured periodic pre-term refreshes as *rejuvenation*.

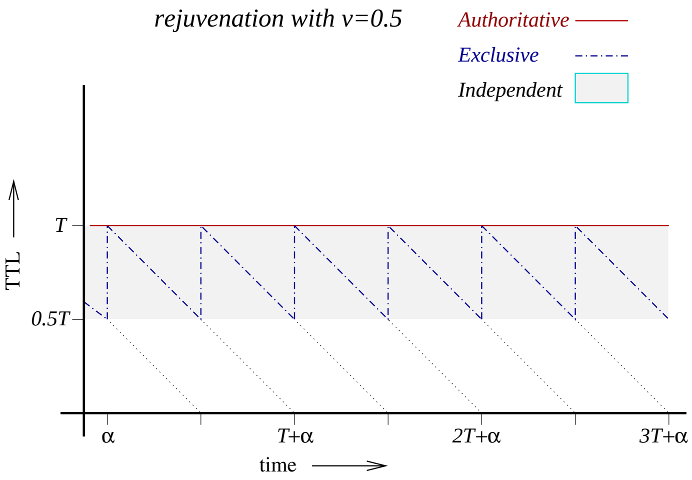

Figure 5: TTL as a function of time for rejuvenating sources with $v = 0.5$. $\text{EXC}_{0.5}$ has a period of $0.5T$ where the age cycles from $T/2$ to T and thus, the TTL cycles from T to $T/2$. For $\text{IND}_{0.5}$ the age at any point in time is independently drawn from $U[T/2, T]$ and thus the TTL is drawn from $U[T/2, T]$. The dotted lines correspond to expiration times of previously-served copies by $\text{EXC}_{0.5}$. The expiration times are at $iT/2 + \alpha$ for integral values of i.

We extend the definitions of our basic sources to include rejuvenation. The source EXC_v is an EXC source that refreshes its copy of the object when the age exceeds v fraction of the lifetime value. Formally, let α be drawn from $U[0, vT]$. At time t, an EXC_v source return the object with age $(t-\alpha) \bmod (v*T)$ (so the TTL is $T-(t-\alpha) \bmod (v*T)$). As with an EXC source, α is fixed for a "run", and performance is the expected performance over runs with different displacements. We say that a client-cache uses an IND_v source if upon each miss it forwards the request to a different independent EXC_v source. Hence, IND_v source returns copies with age drawn from $U[0, vT]$ and thus TTL drawn from $U[(1-v)T, T]$. The TTL as a function of time for the different sources is illustrated in Figure 5. For both IND_v and EXC_v sources, a rejuvenation interval of $v = 1$ corresponds to the respective pure source: $\text{EXC}_1 \equiv \text{EXC}$ and $\text{IND}_1 \equiv \text{IND}$. A rejuvenation interval of $v = 0$ corresponds to a pure AUTH source. That is, $\text{EXC}_0 \equiv \text{AUTH}$ and $\text{IND}_0 \equiv \text{AUTH}$.

Intuitively, we might expect a monotonic improvement in miss rate as v decreases from $v = 1$ to $v = 0$. We show later on that this is the case with IND_v sources but not with EXC_v sources. Note that a client-cache is synchronized with its EXC_v source if and only if $1/v$ is integral.

3 Relationships Between the Basic Sources

In this section we prove some basic relationships between the performances of a client cache through the different sources that we defined. We first prove that on any request sequence the miss rate of a client-cache through an AUTH source is no greater than the miss rate of the client-cache through an EXC source. We also show that for any request sequence the miss rate of a client-cache through an EXC source is no greater than the miss rate of the client-cache through an IND source. To prove this, we first establish two basic lemmas relating the miss rate at a client cache to the distribution of ages (and therefore TTLs) provided by the source. These lemmas will be also useful in later sections.

Lemma 3.1 *Consider two consistent sources s_1 and s_2 such that at any given point in time, the TTL available from s_1 is at least as large as the TTL available from s_2. Then, for any sequence of requests, the number of misses through s_1 is at most the number of misses through s_2.*

Proof. The proof is by induction on the length of the request sequence. The lemma clearly holds for sequences of lengths 1 and 2. Consider a request sequence τ of length l. If the last request is either a miss or a hit through both s_1 and s_2, or a miss through s_2 but a hit through s_1, then the lemma follows by using the induction hypothesis on the prefix of length $l-1$ of τ.

Consider the case where the last request is a miss through s_1 but a hit through s_2. Let the ith request be the last request on which the client working through s_2 had a miss. The client working through s_1 must have had a hit on the ith request and all the requests following it including the next to last request. (Otherwise our assumptions guarantee that it would have got a TTL that covers the last request.) Thus by applying the induction hypothesis on the prefix of τ of length $i-1$ the lemma follows. ■

Lemma 3.2 *Consider two sources s_1 and s_2 that serve objects with TTLs that are independently drawn from probability distributions with CDFs (Cumulative Distribution Functions) F_1 and F_2, respectively. Suppose that for all $x \geq 0$, $F_1(x) \leq F_2(x)$, i.e. s_1 provides longer TTLs. Then for any sequence of requests, the expected number of misses through the source s_1 is no larger than the expected number of misses through the source s_2.*

Proof. We shall use the following claim. For a fixed source s that provides TTLs from a fixed probability distribution with CDF F, and a request sequence τ, the expected number of misses of a client-cache working through s on a suffix τ' of τ is no larger than its expected number of misses on τ. We prove this claim by induction on the length of τ. Let m^i be the expected number of misses on the suffix of τ containing requests $i, i+1, \ldots, \ell$, where ℓ is the length of τ. The number of misses on τ is $1+\sum_{i=2}^{\ell}(F(t_i-t_1)-F(t_{i-1}-t_1))m^i$. Similarly, if τ' is the suffix of τ which starts at the j^{th} request then the expected number of misses on τ' is

$$1 + \sum_{i=j+1}^{\ell} (F(t_i - t_j) - F(t_{i-1} - t_j))m^i \ . \tag{1}$$

By induction $m^i \geq m^{i+1}$ for $i \geq 2$, and therefore we have that

$$F(t_i - t_j)m^i \leq F(t_i - t_1)m^i + (F(t_i - t_j) - F(t_i - t_1))m^{i+1} \ . \tag{2}$$

We use Equation (2) to substitute for $F(t_i - t_j)m^i$ in Equation (1), for every $j+1 \leq i < \ell$, and obtain that the miss rate on τ' is bounded by

$$\begin{aligned}
&1 + \sum_{i=j+1}^{\ell} (F(t_i - t_j) - F(t_{i-1} - t_j))m^i = \\
&\quad 1 + F(t_\ell - t_j)m^\ell + \sum_{i=j+1}^{\ell-1} F(t_i - t_j)m^i - \sum_{i=j+1}^{\ell} F(t_{i-1} - t_j)m^i \leq \\
&\quad 1 + F(t_\ell - t_j)m^\ell + \sum_{i=j+1}^{\ell-1} F(t_i - t_1)m^i + \sum_{i=j+1}^{\ell-1} F(t_i - t_j)m^{i+1} - \sum_{i=j+1}^{\ell-1} F(t_i - t_1)m^{i+1} - \\
&\qquad \sum_{i=j+1}^{\ell} F(t_{i-1} - t_j)m^i =
\end{aligned}$$

$$1 + F(t_\ell - t_j)m^\ell + \sum_{i=j+1}^{\ell-1} F(t_i - t_1)m^i - \sum_{i=j+1}^{\ell-1} F(t_i - t_1)m^{i+1} =$$

$$1 + F(t_{j+1} - t_1)m^{j+1} + (F(t_\ell - t_j) - F(t_{\ell-1} - t_1))m^\ell + \sum_{i=j+2}^{\ell-1} (F(t_i - t_1) - F(t_{i-1} - t_1))m^i$$

Finally since for $j \geq 2$

$$F(t_{j+1} - t_1)m^{j+1} \leq \sum_{i=2}^{j+1}(F(t_i - t_1) - F(t_{i-1} - t_1))m^i \ ,$$

and $F(t_\ell - t_j) \leq F(t_\ell - t_1)$ we obtain the claim.

We now turn to the proof of the lemma. The proof is also by induction on the length of the request sequence. The base case where τ is of length 1 or 2 is obvious. Assume τ is of length $\ell > 2$. Let m_1^i and m_2^i be the expected number of misses through s_1 and s_2 respectively on the suffix of τ that starts at request i (including request i). Our definitions imply that the expected number of misses on τ through s_j, for $j = 1$ or $j = 2$ is

$$1 + \sum_{i=2}^{\ell}(F_j(t_i - t_1) - F_j(t_{i-1} - t_1))m_j^i \tag{3}$$

By the claim above we know that $m_2^i \geq m_2^{i+1}$ for every $i \geq 1$. Also recall our assumption that $F_1(t) \leq F_2(t)$. So we obtain that

$$F_1(t_i - t_1)m_1^i \leq F_2(t_i - t_1)m_1^i + (F_1(t_i - t_1) - F_2(t_i - t_1))m_1^{i+1} \ . \tag{4}$$

for $1 \leq i < \ell$. Substituting this upper bound on $F_1(t_i - t_1)m_1^i$ for $2 \leq i < \ell$ in Equation (3) we obtain that

$$\begin{aligned}
&1 + \sum_{i=2}^{\ell}(F_1(t_i - t_1) - F_1(t_{i-1} - t_1))m_1^i \leq \\
&\quad 1 + F_1(t_\ell - t_1)m_1^\ell + \sum_{i=2}^{\ell-1} F_1(t_i - t_1)m_1^i - \sum_{i=2}^{\ell} F_1(t_{i-1} - t_1)m_1^i \leq \\
&\quad 1 + F_1(t_\ell - t_1)m_1^\ell + \sum_{i=2}^{\ell-1} F_2(t_i - t_1)m_1^i + \sum_{i=2}^{\ell-1} F_1(t_i - t_1)m_1^{i+1} - \sum_{i=2}^{\ell-1} F_2(t_i - t_1)m_1^{i+1} - \sum_{i=2}^{\ell} F_1(t_{i-1} - t_1)m_1^i = \\
&\quad 1 + F_1(t_\ell - t_1)m_1^\ell + \sum_{i=2}^{\ell-1} F_2(t_i - t_1)m_1^i - \sum_{i=2}^{\ell-1} F_2(t_i - t_1)m_1^{i+1} = \\
&\quad 1 + F_1(t_\ell - t_1)m_1^\ell + \sum_{i=2}^{\ell-1} F_2(t_i - t_1)m_1^i - \sum_{i=3}^{\ell} F_2(t_{i-1} - t_1)m_1^i = \\
&\quad 1 + (F_1(t_\ell - t_1) - F_2(t_{\ell-1} - t_1))m_1^\ell + \sum_{i=2}^{\ell-1}(F_2(t_i - t_1) - F_2(t_{i-1} - t_1))m_1^i
\end{aligned}$$

Finally we obtain the lemma using the fact that by assumption $F_1(t_\ell - t_1) \leq F_2(t_\ell - t_1)$, and the induction hypothesis saying that $m_1^i \leq m_2^i$ for $i \geq 2$. ■

As a consequence of Lemma 3.1 we obtain that AUTH has smaller miss rate than EXC. Furthermore, as a consequence of Lemma 3.2 we obtain that AUTH has smaller miss rate than IND. The following Lemma establishes that EXC always outperforms IND.

Lemma 3.3 *For any sequence of requests, the miss rate through an* EXC *source is never higher than through an* IND *source.*

Proof. Consider the ith request. Its likelihood of being a miss with EXC is $\min\{1,(t_i - t_{i-1})/T\}$. Let $m[j]$ and $h[j]$ denote the events that the jth request is a miss or a hit respectively, with an IND source. The likelihood that the ith request constitutes a miss with IND is

$$\begin{aligned} p(m[i]) &= p(m[i-1])\min\{1,\frac{t_i - t_{i-1}}{T}\} \\ &\quad +p(h[i-1]\cap m[i-2])\min\{1,\frac{t_i - t_{i-2}}{T}\} \\ &\quad +p(h[i-1]\cap h[i-2]\cap m[i-3])\min\{1,\frac{t_i - t_{i-3}}{T}\}+\ldots \\ &\geq \min\{1,\frac{t_i - t_{i-1}}{T}\}\ . \end{aligned}$$

Since the likelihood of every request to be a miss through an EXC source is smaller than through an IND source, we obtain that the miss rate through EXC is smaller than the miss rate through IND. ■

Note that with respect to one fixed displacement value α, the EXC source could perform worse than IND. Consider the following sequence where requests are made at times $(2i-\epsilon)T$ and $(2i+\epsilon)T$ for integral $i>0$. Suppose that the source refreshes at times iT. Then all requests would constitute misses with EXC, and only $1/2+2\epsilon$ would constitute misses with IND. Lemma 3.3 shows that on average over all displacements, EXC performs at least as well as IND.

Lemma 3.3 has interesting implications for configuring a set of top-level web content caches to serve a population of clients. It shows that we benefit from configuring a client to send all its requests, or at least all its requests for a particular object, to the same primary cache. Configuring this way we obtain that for every object the source is EXC rather than a hybrid with IND. The miss-rate at the high-level caches, however, is likely to be smaller and more stable if the workload is partitioned by object. This partition maximizes the number of clients that can benefit from the object being cached. To summarize, our analysis shows that it is best if the client, when it has a choice, picks a high-level cache according to the requested object.

4 Rejuvenations and Extensions

In this section we show that there is a relationship between rejuvenating sources (EXC_v sources) and clients extending the freshness lifetime of the object ($\text{EXC}(r)$ sources). More specifically, we show that for each sequence of requests there is a related sequence such that the miss rate of EXC_v on the first sequence is the same as the miss rate of $\text{EXC}(r=\frac{1}{v})$ on the second sequence. We also show that for high request rate there is no point for an $\text{EXC}(r)$ source to use a nonintegral extension factor. Finally we use the relationship between the sources to show that rejuvenation may degrade performance and suggest ways to avoid such degradation.

We denote by $\{t_i\}$ a sequence of requests at times $t_1, t_2, \ldots, t_n$. We shall also refer to the ith request as *request* t_i. We prove the following theorem.

Theorem 4.1 *Consider the sources* EXC_v*, and* $\text{EXC}(r)$*, for* $r = \frac{1}{v}$*. Then the miss rate of a client-cache through* EXC_v *on a request sequence* $\{t_i\}$ *is identical to the miss rate of a client-cache through* $\text{EXC}(r)$ *on the sequence* $\{rt_i\}$*. The same is true for* IND *sources.*

Proof. First we show that for each displacement $\alpha \in [0, vT]$ of EXC_v there is a unique displacement $\beta = r\alpha$ of $\text{EXC}(r)$ ($r = 1/v$) such that the miss rate on $\{t_i\}$ through EXC_v with displacement α is the same as the miss rate on $\{rt_i\}$ through $\text{EXC}(r)$ with displacement β.

Let $\alpha_j = (t_j - \alpha) \bmod (vT)$, and let $\beta_j = (rt_j - \beta) \bmod T$. It is easy to check that $\beta_j = r\alpha_j$. We now prove by induction on the length of the request sequence that there is a miss at time t_j using the source EXC_v if and only if there is a miss at time rt_j using the source $\text{EXC}(r)$. The base of the induction is obviously true. Assume that for every $\ell \leq j-1$, t_ℓ is a miss using the source EXC_v iff rt_ℓ is a miss using the source $\text{EXC}(r)$. Consider the jth request, and let $k \leq j-1$ be the index such that t_k and rt_k are misses of EXC_v and $\text{EXC}(r)$ respectively but t_i and rt_i are hits for every $k < i < j$.

The request at time t_j is a miss through EXC_v if and only if $t_j - t_k + \alpha_k \geq T$. The request at time rt_j is a miss through $\text{EXC}(r)$ if and only if $rt_j - rt_k + \beta_k \geq rT$. Since $\beta_k = r\alpha_k$ we obtain that t_j is a miss for EXC_v iff rt_j is a miss for $\text{EXC}(r)$.

Since the linear mapping from α to β is density-preserving we obtain that the correspondence of the miss rates with respect to fixed displacements implies a correspondence of the expected performance across all displacements. Hence, the theorem follows.

A similar proof holds for IND sources. ■

Notice that Theorem 4.1 does not imply that extensions and rejuvenations are equivalent effects, as the miss rates are measured for different sequences. For example, applying this correspondence to two instances with the same sequence and sources EXC_v and EXC_u, we obtain two instances with two different request sequences (and two different extension factors).

4.1 Does Lifetime Extension Pay Off?

We first observe that generally, extending an object's lifetime cannot increase the miss rate. We then show that sometimes, however, extending the lifetime does not decrease the miss-rate so as a result we increase staleness without a corresponding gain.

Lemma 4.1 *Consider two client caches* c_1 *and* c_2 *that both use either* AUTH*,* EXC*, or* IND *sources. Suppose that upon each miss* c_1 *uses lifetime of* $r_1 * T$ *and* c_2 *uses lifetime of* $r_2 * T$ *where* $r_2 \geq r_1 \geq 1$*. Then the miss-rate of* c_2 *is at most that of* c_1*.*

Proof. The proof for AUTH follows from Lemma 3.1. The proof for EXC with respect to any fixed displacement follows from Lemma 3.1, and still holds when we take the expectation over all displacements. The claim for IND source follows from Lemma 3.2. ■

The following Lemma shows that we may not reduce the miss rate for some frequently requested objects by increasing the extension factor to a non integral value. Therefore in such cases we increase staleness without a corresponding gain in miss rate.

Lemma 4.2 *Consider an* $\text{EXC}(r)$ *source and a request sequence at the client cache such that the object is requested at least once every* $(\lceil r \rceil - r)T$ *time units. Then, the miss rate of the client-cache on such sequence is the same as with* $\text{EXC}(\lfloor r \rfloor)$*,*

Proof. When the object expires at the subsidiary cache, the copy at the source has age $(r - \lfloor r \rfloor)T$. Since the object is requested at least once every $(\lceil r \rceil - r)T$ time units it must be requested again before the parent refreshes its own copy. Thus, a miss at the client cache is incurred once every $\lfloor r \rfloor T$ time units. ■

For low request rates, however, non-integral extension factors could be beneficial even with EXC(r). For example, if an object is requested with intervals of $1.2T$, then the miss rate would be 1 with extension factor $r \leq 1.2$ and would be 0.8 with $r = 1.5$.

4.2 Does Rejuvenating Pay Off?

At first glance, it seems that rejuvenations can only improve performance at the client cache. Somewhat surprising is the fact that this is not the case for EXC rejuvenating sources. Using Theorem 4.1 and Lemma 4.2, we can prove the following lemma.

Lemma 4.3 *Let $v < 1$, and let $u = \frac{1}{\lfloor 1/v \rfloor}$. Note that $v \leq u \leq 1$. Consider a request sequence $\{t_i\}$, such that the object is requested at least once every $(\lceil \frac{1}{v} \rceil v - 1)T$ time units. Let m_v denote the miss rate through* EXC$_v$ *on $\{t_i\}$ and m_u the miss rate through* EXC$_u$ *on $\{t_i\}$. Then, $m_v = \dfrac{u}{v} m_u$.*

Proof. First, let $r = \frac{1}{v}$. By Theorem 4.1, m_v equals the miss rate through EXC(r) on the sequence $\{rt_i\}$. Since in $\{rt_i\}$ there is a request every $(\lceil \frac{1}{v} \rceil - \frac{1}{v})T = (\lceil r \rceil - r)T$ times units then using Lemma 4.2 we obtain that the miss rate of $\{rt_i\}$ through EXC(r) is equal to its miss rate through EXC$(\lfloor r \rfloor)$. We claim that the miss rate through EXC$(\lfloor r \rfloor)$ on $\{rt_i\}$ equals $\frac{r}{\lfloor r \rfloor}$ times the miss rate through EXC$(\lfloor r \rfloor)$ on $\{\lfloor r \rfloor t_i\}$. Assuming this claim is established then by applying Theorem 4.1 again the theorem follows.

To prove the claim, notice that by the terms of the lemma, for every $i \geq 1$, $t_{i+1} - t_i < (\lceil \frac{1}{v} \rceil v - 1)T$, and thus, $rt_{i+1} - rt_i < (\lceil r \rceil - r)T < T$. Therefore, when using an EXC$(\lfloor r \rfloor)$ source, for both the sequence $\{rt_i\}$, and the sequence $\{\lfloor r \rfloor t_i\}$ we have a miss in every T-interval where the extended lifetime ends, i.e. once in every $\lfloor r \rfloor T$ time units. From this follows that the miss rate of EXC$(\lfloor r \rfloor)$ on each of these two sequences equals to one over the average number of requests in an interval of length $\lfloor r \rfloor T$ in the corresponding sequence. Since the sequences $\{rt_i\}$ and $\{\lfloor r \rfloor t_i\}$ contain the same number of requests and the time span of $\{rt_i\}$ is larger by a factor of $\frac{r}{\lfloor r \rfloor}$ than the time span of $\{\lfloor r \rfloor t_i\}$ we obtain that the average number of requests of $\{rt_i\}$ in any fixed length interval is smaller by a factor of $\frac{\lfloor r \rfloor}{r}$ than the average number of requests of $\{\lfloor r \rfloor t_i\}$ in the same interval so the claim follows. ■

Lemma 4.3 demonstrates that the performance of a cache which receives frequent requests to the object can be strictly worse through EXC$_v$ with $v < 1$ than through a non-rejuvenating EXC source. For example, for $v > 0.5$, and for sequences satisfying that the object is requested at least once every $(2v - 1)T$ time units, by Lemma 4.3 the miss rate is strictly worse through EXC$_v$ than with a non-rejuvenating EXC source. In this case, rejuvenating does not pay off.

In contrast, the following lemma shows that IND sources do exhibit monotonic dependence of the miss rate on v.

Lemma 4.4 *Let* IND$_{v_1}$ *and* IND$_{v_2}$ *be two sources such that $v_1 \leq v_2$. Then, the miss rate through* IND$_{v_1}$ *is no larger than the miss rate through* IND$_{v_2}$*.*

Proof. By the definition of IND$_v$ the TTL's through IND$_{v_1}$ are drawn from $U[(1 - v_1)T, T]$ and the TTL's through IND$_{v_2}$ are drawn from $U[(1 - v_2)T, T]$. Let F_1 be the CDF of $U[(1 - v_1)T, T]$ and let F_2 be the CDF of $U[(1-v_2)T, T]$. Since for all $x \geq 0$, $F_1(x) \leq F_2(x)$ Lemma 3.2 implies that the miss rate through IND_{v_1} is no larger than the miss rate through IND_{v_2}. ∎

The following corollary of Lemma 3.1 shows that although generally rejuvenation does not always improve the performance, rejuvenation cannot degrade performance on any sequence if the source preserves synchronization.

Corollary 4.1 *Suppose a rejuvenating* EXC *source adheres to the original refresh schedule, refreshing the object at times $\alpha + iT$ for integral i in addition to possibly rejuvenating it at other points in time. Then on any sequence of requests, the number of misses is not higher than the number of misses through* EXC.

Proof. Follow-up rejuvenations guarantee that at any given point in time, the TTL obtained from the source with rejuvenation is at least as large as the TTL obtained without rejuvenation. The proof then follows by Lemma 3.1. ∎

In particular it follows from Corollary 4.1 that the performance through EXC_v with integral $1/v$, (i.e., $v = 1/2, 1/3, \ldots$), is at least as good as through EXC. In Section 6.2 we show that even if we restrict ourselves to EXC_v, with integral $1/v$, the miss rate is not always monotonic as a function of v, and moreover, there are examples where EXC_v performs worse than an IND source.

5 Worst-Case Performance

In this section we analyze the worst case performance of EXC and IND rejuvenating sources. For a source S and a request sequence $\{t_i\}$ let $\rho(\{t_i\}, S)$ be the ratio between the expected miss rate of $\{t_i\}$ through S and through an AUTH source. The supermom over all sequences $\{t_i\}$ of $\rho(\{t_i\}, S)$ is called the *competitive ratio* of S, and is denoted by $\rho(S)$. We shall use the term *rejuvenating policy* for any source that corresponds to a single high-level cache which keeps a fresh copy of the object by renewing it (rejuvenating) from an authoritative server before or when it expires. In particular AUTH, EXC, and EXC_v, are rejuvenating policies. Note that a rejuvenating policy is consistent. The next lemma shows that the competitive ratio of any rejuvenating policy is at most 2.

Lemma 5.1 *The competitive ratio of any rejuvenating policy is at most 2.*

Proof. We first establish that the client can get at most two misses in each time interval of length T. Consider two consecutive misses. By the definition of a rejuvenating policy the source must refresh the object and reduce its TTL to zero at least once between the two misses. Therefore the sum of the TTL obtained at the second miss, and the elapsed time between the two misses must be at least T. So the third miss can occur only after at least T time units past the first miss.

Let the miss-times through an AUTH source be $m_1, m_2, \ldots, m_k$. Clearly, the set of all request times is covered by the mutually disjoint intervals $[m_i, m_i + T)$, as a request which is not covered by any of these intervals must be a miss. Through an AUTH source the client has exactly one miss in each of these intervals, while through a rejuvenating policy the client could have at most 2. Thus the competitive ratio is at most 2. ∎

Next we establish a corresponding lower bound of 2 on the competitive ratio of any rejuvenating policy that cannot rejuvenate the object continuously but must have a small gap of ϵ between consecutive rejuvenations.

Theorem 5.1 *Consider a rejuvenating policy P such that the time interval between any two rejuvenation points must exceed some $\epsilon > 0$. Then $\rho(P) = 2$.*

Proof. By Lemma 5.1, we get that $\rho(P) \leq 2$. We first prove the $\rho(P) \geq 2$ when P is deterministic. If P is deterministic then an adversary can construct a request sequence consisting of pairs of requests as follows. The gap between the two requests constituting a pair is T time units. The gap between the second request of a pair and the first request of the following pair is larger than T so the first request of a pair is a miss through P and through AUTH. Furthermore, the first request in each pair shows up ϵ time units after P refreshes the object. Therefore through P a client cache misses on the second request of each pair. Through AUTH however, the second request of each pair is a hit. Therefore $\rho(P) \geq 2$.

For a randomized source P we prove that $\rho(P) \geq 2 - \delta$ for every $\delta > 0$ using Yao's lemma [15]. By Yao's lemma in order to prove this lower bound it suffices to construct a distribution, D, on sequences of requests such that the ratio between the expected miss rate of every deterministic source P (over D) and the expected miss rate of AUTH (over D) is at least $2 - \delta$.

We define D as follows. Let t_1 be chosen uniformly from an arbitrary interval of length ϵ. For every $k \geq 1$, let $t_{2k} = t_{2k-1} + T - \epsilon\delta$, and let t_{2k+1} be chosen from $U[t_{2k} + 2T, t_{2k} + 2T + \epsilon)$. Let P be an arbitrary deterministic policy. Clearly, t_{2k-1} constitutes a miss for both AUTH and P. Using the assumption that the smallest distance between any two rejuvenation points of P is ϵ, it is easy to see that for every $k \geq 1$, the probability over the request times distribution, that the TTL of t_{2k-1} when using P is at least $T - \epsilon\delta$, is at most δ. Therefore the request t_{2k} is a miss of P with probability at least $1 - \delta$, and the expected miss rate of P over D is at least $(2 - \delta)/2$. An AUTH source misses only on the first request of each pair and therefore its miss rate is $\frac{1}{2}$. To summarize we obtain that the ratio of the expected miss rate of P over D and the expected miss rate of AUTH over D is at least $2 - \delta$. ■

In particular, Theorem 5.1 implies that rejuvenating in fixed intervals, (an EXC_v source with $v > 0$), has a competitive ratio 2.

Another related problem is to compare the worst-case performance of an IND_v source with respect to an AUTH source. Since an IND_v source is not consistent it is not a rejuvenating policy and therefore Theorem 5.1 does not apply. To give some intuition, if $v < 1/2$, then the TTL provided for every request is at least $T/2$, and thus, in every interval of length T time units, there are at most 2 misses. Therefore, in this case, the proof for Lemma 5.1 holds, and thus, the competitive ratio of IND_v is at most 2. To see that it is exactly 2, consider a sequence where requests are spaced T time units apart. The miss rate is 1 through IND_v source whereas it is 1/2 through an AUTH source. We obtain the same ratio of 2 on a sequence of requests that is constituted of subsequences, where subsequences are spaced more than T apart and in each subsequence the distance between the last request and the first request is exactly T. More generally, we prove the following theorem.

Theorem 5.2 *The competitive ratio of* IND_v *is* $\displaystyle\sum_{k=0}^{\lfloor \frac{1}{1-v} \rfloor} \frac{(1-k(1-v))^k}{k! v^k}$.

Thus, by Theorem 5.2, for $v < 1/2$, the competitive ratio of an IND_v source is 2, and as v approaches 1, the competitive ratio approaches $e \approx 2.718$. The higher competitive ratio of IND_v comparable to the competitive ratio of EXC_v fits well with the superiority of EXC on IND established in Section 3.

Let $X_i \sim U[T(1-v), T]$, and let Y_v be a random variable taking the value k, where k is the minimum such that $X_1 + X_2 + \cdots + X_k \geq T$. In order to prove Theorem 5.2, we need the following two lemma.

Lemma 5.2 *The competitive ratio of* IND_v *equals the expectation of* Y_v.

Proof. For the lower bound, consider a set of requests, such that the time difference between the last request and the first request in the set is exactly T, and the difference between any two consecutive requests is $\frac{1}{n}$. Assemble a sequence consisting of many such sets of requests where there are at least T time units between the last request in one set and the first request of the following set.

An AUTH source will only have one miss per each set of requests spanning a T-interval in the sequence, while the expected number of misses that an IND_v source will have approaches $E[Y_v]$ per interval, when n goes to infinity. Thus, we get that the competitive ratio of IND_v is at least $E[Y_v]$.

For the upper bound, consider the misses $m_1, \ldots, m_k$ of the AUTH source. Then, the intervals $[m_i, m_i + T)$ are mutually disjoint, and cover the sequence of requests. The expected number of misses in each such interval is at most $E[Y_v]$, and therefore, we get that this is an upper bound for the competitive ratio of IND_v. ■

As a side note, we remark that for any source which draws TTLs independently from some distribution $R[0, T]$ a Lemma analogous to Lemma 5.2 holds. In such lemma the definition of Y_v should use $X_i \in R[0, T]$ rather $X_i \in U[(1-v)T, T]$.

The next lemma shows that the expectation of Y_v is indeed as claimed in Theorem 5.2.

Lemma 5.3 *For every $v \leq 1$,*

$$E[Y_v] = \sum_{k=0}^{\lfloor \frac{1}{1-v} \rfloor} \frac{(1 - k(1-v))^k}{k! v^k}.$$

Proof. Let $Z_i \sim U[0, 1]$, and let $P(k, a)$ be the probability that $Z_1 + \ldots + Z_k < a$, where $1 \geq a \geq 0$, and $k \geq 1$ is an integer. We show by induction that $P(k, a) = \frac{a^k}{k!}$. Clearly, $P(1, a) = \int_0^a dx = a$. Moreover,

$$P(k, a) = \int_0^a P(k-1, a-x)dx = \int_0^a \frac{(a-x)^{k-1}}{(k-1)!}dx = \frac{a^k}{k!}.$$

Now, let $X_i \sim U[T(1-v), T]$, and let Y_v be as defined above. Let $Z_i = (X_i - (1-v)T)/(Tv)$. Clearly, $Z_i \sim U[0, 1]$. The probability that $Y_v \geq k+1$ is exactly the probability that $X_1 + \ldots X_k < T$. Clearly, $X_1 + \ldots + X_k < T$ if and only if $Z_1 + \ldots Z_k < k - (k-1)/v$. If $k > 1/(1-v)$, then $k - (k-1)/v < 0$, and therefore, the probability that $Y_v \geq k+1$ is zero. Otherwise,

$$P(Y_v \geq k+1) = P(k, k - \frac{k-1}{v}) = \frac{(1 - k(1-v))^k}{v^k k!}.$$

Therefore, the expectation of Y_v is

$$\begin{aligned} E[Y_v] &= \sum_{k=1}^{\infty} P(Y_v \geq k) \\ &= 1 + \sum_{k=1}^{\lfloor \frac{1}{1-v} \rfloor} \frac{(1 - k(1-v))^k}{v^k k!} = \sum_{k=0}^{\lfloor \frac{1}{1-v} \rfloor} \frac{(1 - k(1-v))^k}{v^k k!}. \end{aligned}$$

■

The worst-case analysis performed in this section does not distinguish between the performance of sources with different rejuvenation rates, as all EXC_v sources $(0 < v \leq 1)$ and all IND_v sources $(0 < v \leq 1/2)$ have the same tight competitive ratio of 2. Next, we analyze specific distributions and obtain more refined relationships.

6 Requesting an object with known distribution

So far we presented relationships between various sources that hold for every request sequence. In this section we calculate precisely the miss rate of the sources for particular sequences. We consider two kinds of sequences, 1) sequences where inter-request times are independently drawn from a Poisson distribution, and 2) sequences with fixed inter-request time between requests. We consider these two families of sequences since they model real world phenomenon. Poisson inter-request times model *reference locality*, which means that the next request is more likely to occur sooner. Fixed interrequest intervals model sequences generated automatically or by robots. Actual inter-request time histogram of an object is likely to be a mix of fixed-frequency arrivals with another distribution which posses reference-locality. Obviously Poisson distribution is not the only distribution which models reference locality, and probably a Pareto distribution which has a heavy tail is better suited for modeling web access sequences [10]. The advantage of the Poisson distribution is that it allows easy analytical analysis. Experimental results regarding the Pareto distribution and real web sequences are presented in [4].

6.1 Poisson arrivals

Here we consider request sequences defined such that the density function of inter-request durations is $f(x) = \frac{\lambda}{T}e^{-\frac{\lambda}{T}x}$ and the respective CDF (likelihood for an inter-request duration not to exceed x) is $F(x) = 1 - e^{-\frac{\lambda}{T}x}$. For these sequences the average number of requests in a time duration of length 1 (the inverse of the average inter-request duration) is λ/T. This quantity is also called the *rate* of the process. Consequently there are λ requests per lifetime interval T.

We analyze the expected miss rate at a client cache which uses extension factor of r (when $r = 1$ this is a basic source) on such sequences. We calculate the miss rate as a function of r and λ for the different source types. For simplicity, we write $r = k + x$, where $k = \lfloor r \rfloor$, and $0 \leq x < 1$. We prove the following lemma.

Lemma 6.1 *The miss rate of a request sequence generated by a Poisson process with rate λ/T as defined above is:*

- $\frac{1}{1+r\lambda}$ *through an* AUTH(r) *source.*
- $\frac{1}{1+(r-1/2)\lambda}$ *through an* IND(r) *source.*
- $$\frac{1}{\lambda\left(k + \frac{e^{\lambda x}}{e^{\lambda}-1}\right)}$$ *through an* EXC(r) *source.*

Proof. The miss rate through an AUTH(r) source is determined by the expected number of requests in an rT-interval following a cache miss. If this expected number of requests is β, then the miss rate is $1/(1+\beta)$. For Poisson arrivals the expected number of requests in any interval is proportional to the interval length. Therefore the expected number of requests in an interval of length rT is $r\lambda$ and the miss rate through AUTH is $1/(1 + r\lambda)$.

For an IND(r) source, the miss rate depends on the expected number of requests occurring after a miss in an interval drawn from $U[(r-1)T, rT]$. If this number is β, then the miss rate is $1/(1+\beta)$. Since the expected length of such interval is $(r - 1/2)T$ we obtain that the expected number of requests in an interval drawn from $U[(r - 1)T, rT]$ is $(r - 1/2)\lambda$. It follows that the miss rate of IND(r) is $\frac{1}{1+(r-1/2)\lambda}$.

We now calculate the miss rate through an EXC(r) source. We partition time into T-intervals according to the EXC source refreshment cycle. We consider the following partition of time into coarser intervals. Each coarse interval is a continuous range of T-intervals. Every cache miss has a corresponding coarse interval that starts with the T-interval that includes the miss and ends just before the interval that includes the next miss. We now compute the expected size (in number of T-intervals) of each coarse interval.

Each coarse interval is of length at least k, by definition of the extension factor. It is of length exactly k if a request arrives in the $(k+1)^{th}$ T-interval after the extended TTL expires. The latter has probability $(1-e^{-\lambda(1-x)})$. Therefore the probability that the length of the coarse interval is at least $k+1$ T-intervals is $e^{-\lambda(1-x)}$. Given that the length of a coarse interval is at least $k+1$ T-intervals, the probability that it is larger than $k+i$, $i \geq 2$, T-intervals is $e^{-\lambda(i-1)}$ (the probability that i empty intervals follow the expiration interval). Thus, the expected length of a coarse interval is

$$\begin{aligned} C(r,\lambda) &= k(1-e^{-\lambda(1-x)}) + e^{-\lambda(1-x)}\left(k+1+\frac{e^{-\lambda}}{1-e^{-\lambda}}\right) \\ &= k + e^{\lambda(x-1)}\left(1+\frac{1}{e^{\lambda}-1}\right) = k + \frac{e^{\lambda x}}{e^{\lambda}-1} \end{aligned}$$

The rate of the misses per T-interval is $1/C$. Since the expected number of requests in a T-interval is λ, the miss rate is $1/(\lambda C)$. ■

The result proved in Lemma 6.1 is visualized in Figure 6 where the miss rate as a function of r is plotted for the three sources. For AUTH and IND sources the tradeoff curve is convex and monotonically decreasing. For EXC sources the tradeoff curve of miss rate vs. extension is concave between any two integral values of r, but is convex and monotone decreasing when restricted to integral values of r. Notice that for high rates (large values of λ), the miss rate of EXC(r) is $\approx 1/(\lambda\lfloor r\rfloor)$, which is a step-like function with drops at integral values of r. The figure also shows that across all request rates, EXC(r) outperforms IND(r) for integral r's but is outperformed by IND(r) on some intermediate values.

In practice the selection of the extension factor can vary with object and with time, as to obtain a better balance of the *degree of incoherence* and the *cost of cache misses*. Let us assume that for a particular object, the *degree of incoherence* is the average value of the extension factor r, and the *cost of cache misses* is proportional to the number of misses. We now ask, given maximum allowed incoherence, how to select extension factors as to minimize the miss-cost. The form of the miss rate vs. extension tradeoff curve of EXC(r) implies that across all request rates, integral values of r provide the best tradeoff of incoherence and miss-rate. In particular, a mix of consecutive integral values of the extension factor incurs lower miss-cost than a non-integral extension factor with the same degree of incoherence. (A mix of EXC(1) and EXC(2) with the same incoherence as EXC(1.5) has lower cost.) However, with IND and AUTH sources, the dependence of miss-rate on the extension factor is convex, and monotone decreasing as r increases. Thus, a fixed value of r would result in lower miss-rate than a mix of values with the same average degree of incoherence.

We now consider rejuvenating sources. Let $v < 1$, and let $r = 1/v, r = k + x$, where $k = \lfloor r \rfloor$, and $0 \leq x < 1$. As a corollary of Lemma 6.1 and Theorem 4.1, we get the following lemma:

Lemma 6.2 *The miss rate of a request sequence generated by a Poisson process with rate λ is:*

- $\frac{2}{2+\lambda(2-v)}$ *through an* IND$_v$ *source.*

- $\dfrac{1}{\lambda v\left(k+\frac{e^{\lambda v x}}{e^{\lambda v}-1}\right)}$ *through an* EXC$_v$ *source.*

Figure 6 shows the miss-rate dependence on the rejuvenation interval. The value $1/v$ can be viewed as the "overhead" of the rejuvenating source. Lemma 6.2 reiterates that integral values of $1/v$ are more effective, as non-integral values of $1/v$ often result in "negative" utility, where the miss-rate increases with overhead. For Poisson and integral $1/v$ values, we observe a desirable tradeoff where the miss-rate is monotonic decreasing with the overhead. Also observe that for integral $1/v$, EXC$_v$ is more effective than IND$_v$.

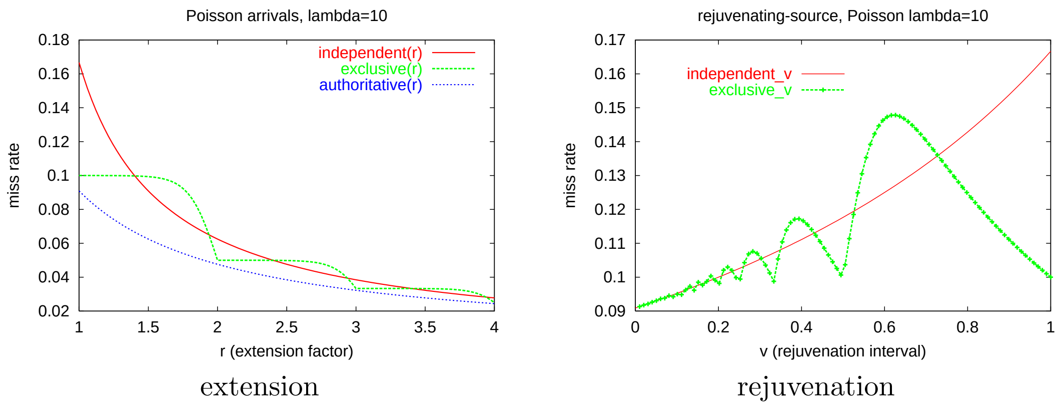

Figure 6: Miss-rate of different source-types on Poisson requests with rate $\lambda = 10$ with varying rejuvenation intervals or extension factors.

6.2 Requesting an object with fixed frequency

In this section we consider a client-cache that gets requests for an object with fixed interrequest interval of $f * T$ $(0 < f)$ where T is the object's lifetime. The following lemma specifies the miss rate of the client cache through EXC(r) when r is an integer for almost every possible f.

Lemma 6.3 *Let r be an integer. For a request sequence with fixed inter-request times of $f * T$ time units, the miss rate through an* EXC(r) *source is:*

- *When $f \leq 1$, the miss rate is $\frac{f}{r}$.*
- *When $\frac{r}{m+1} < f \leq \frac{r-1}{m}$, where m is an integer satisfying $m+1 < r$, the miss rate is $\frac{1}{m+1}$.*
- *When $f = r - 1 + x$, where $\frac{1}{2k+1} < x \leq \frac{1}{2k}$, for $k \geq 1$, the miss rate is $1 - kx$.*
- *When $f = r - 1 + x$, where $\frac{1}{2k} < x \leq \frac{1}{2k-1}$, for $k \geq 1$, the miss rate is kx.*
- *When $f > r$, the miss rate is 1.*

Proof. When $f < 1$, we have a miss every interval of length rT. Therefore the miss rate equals to one over the average number of requests in an interval of length rT which is $\frac{f}{r}$.

When $\frac{r}{m+1} < f \leq \frac{r-1}{m}$, where $m < r - 1$, after each miss at time t_i, the request t_{i+m+1} will be in time $t_i + (m+1)fT > t_i + rT$, and therefore, it is going to be a miss. On the other hand, the request t_{i+j}, for $j \leq m$, will be in time $t_i + jfT \leq t_i + (r-1)T$, and therefore will be a hit. Thus, the miss rate in this case is $\frac{1}{m+1}$.

Let $f = r - 1 + x$. We define a request t_i as *special* if $(t_i - \alpha) \bmod T \in [0, xT)$ where α is the displacement of the source. By the definition of $\text{EXC}(r)$, $\alpha \in U[0, T]$, so the probability that t_i is special is x for every i. By the definition of $\text{EXC}(r)$ every special request is a miss. The requests in between two consecutive special requests alternate with the first request after the first special request being a hit the next one is a miss and so on until the request preceding the following special request.

We associate every miss with the first special request preceding it. We will calculate the average (over the choices of α) of the number of misses associated with each request. It is easy to see that this average is exactly the miss rate.

Assuming α is such that t_i is special, the number of requests between t_i and the following special request (including the first special request but not the second) is either $\lfloor \frac{1}{x} \rfloor$ or $\lceil \frac{1}{x} \rceil$. It is $\lceil \frac{1}{x} \rceil$ if an only if $(t_i - \alpha) \bmod T \in [0, (1 - \lfloor \frac{1}{x} \rfloor x)T)$. Assuming t_i is special this event happens with probability $(1 - \lfloor \frac{1}{x} \rfloor x)/x$.

Let $\frac{1}{2k+1} < x \leq \frac{1}{2k}$. The argument above shows that the number of requests between t_i and the following special request is $2k+1$ with probability $(1-2kx)/x$ and $2k$ with probability $1 - (1-2kx)/x$. In case there are $2k+1$ requests in between the two, then $k+1$ of them are misses, and in case there are k requests in between the two, only k of them are misses. Therefore the average number of misses associated with t_i assuming t_i is special is

$$(k+1)\frac{1-2kx}{x} + k\frac{x-(1-2kx)}{x} = \frac{1-kx}{x}.$$

Multiplying this by x, which is the probability that t_i is special, we obtain that the average number of misses associated with any request is $1 - kx$. This is also the miss rate.

Let $\frac{1}{2k} < x \leq \frac{1}{2k-1}$ then the argument above shows that the number of requests between t_i and the following special request is either $2k-1$ or $2k$. In either case k of these requests are misses. So assuming t_i is special there are k misses associated with it. Multiplying by the probability that t_i is a miss we obtain that the miss rate in this case is kx.

When $f > r$, then every request is a miss, and thus, the miss rate is 1. ■

As a consequence of Lemma 6.3, we get that the miss rate through EXC_v source is not monotonic as a function of v, even if we consider only v's such that $\frac{1}{v}$ is integral. For example, take a sequence of fixed frequency request times with $f = 3/4 + y$, where $0 < y \leq 1/36$. By Theorem 4.1, for $v = 1/2$, the miss rate is the same as for a sequence of requests at regular intervals with $f = 3/2 + 2y$, through an $\text{EXC}(2)$ source. Hence, by Lemma 6.3, we get that for $v = \frac{1}{2}$ the miss rate is $1/2 + 2y$. For $v = 1/3$, the miss rate is the same as for a sequence of requests at regular intervals with $f = 9/4 + 3y$, through an $\text{EXC}(3)$ source. Thus, by Lemma 6.3, for $v = \frac{1}{3}$ the miss rate is $1/2 + 6y$. Therefore, for this sequence we obtain that it is better to use $v = 1/2$ than $v = 1/3$.

7 Conclusion

We considered age-related performance issues for a distributed system of caches that use TTL-based consistency. Our analysis revealed interesting patterns that may seem counter-intuitive. Despite the wide-scale use of TTL-based consistency (for web caches and within the Domain Name System), its performance effects are not properly understood by practitioners [3]. The models we used closely follow the true behavior of web caches and we believe that our results provide insights for increasing the effectiveness of web caches.

Our results are complemented and supported by extensive simulations using Poisson and Pareto arrivals and traces from web content caches [4]. In particular, these results further demonstrate the presence of interesting dependencies of performance on the extension factor or rejuvenation interval values. The patterns which we observe are similar for the heavy-tailed Pareto, Poisson, and for actual request sequences, but yet, are not universal (e.g., does not occur with fixed inter-arrival times). An intriguing open question is thus to characterize the class of request distributions for which these patterns occur.

Our work suggests some guidelines to follow when configuring caches in a hierarchy. here are few examples.

1. Since an EXC source outperforms an IND source we may want our cache to use the same high level cache consistently for each object.

2. If we configure a cache to rejuvenate we may want to keep it synchronized with its client caches.

3. When we configure a cache to use longer lifetime we want to use a freshness lifetime which is an integral multiple of the lifetime used by the source.

References

[1] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol — HTTP/1.0. RFC 1945, 1996.

[2] E. Cohen, E. Halperin, and H. Kaplan. Performance aspects of distributed caches using TTL-based consistency. In *Proc. of the 28th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science vol 2076, pages 744–756. Springer Verlag, 2001.

[3] E. Cohen and H. Kaplan. The age penalty and its effect on cache performance. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, 2001.

[4] E. Cohen and H. Kaplan. Aging through cascaded caches: performance issues in the distribution of web content. In *Proceedings of the ACM SIGCOMM Conference*, pages 41–53. ACM, 2001.

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. RFC 2616, 1999.

[6] Y. Hou, J. Pan, B. Li, X. Tang, and S. Panwar. Modeling and analysis of an expiration-based hierarchical caching system. In *Proceedings of IEEE Globecom Internet Performance Symposium*. IEEE, 2002.

[7] Y. Hou, J. Pan, C. Wang, and B. Li. On prefetching in hierarchical caching systems. In *Proceedings of IEEE ICC Global Services and Infrastructure for Next Generation Networking Symposium*. IEEE, 2003.

[8] Inktomi Traffic Server.
`http://www.inktomi.com`.

[9] IRCache home.
http://www.ircache.net.

[10] J. Jung, A. Berger, and H. Balakrishnan. Modeling TTL-based internet caches. In *Proceedings of the IEEE Infocom*, 2003.

[11] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. Dns performance and the effectiveness of caching. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2001.

[12] P. Mockapetris. Domain names – concepts and facilities. RFC 1034, 1987.

[13] P. Mockapetris. Domain names – implementation and specification. RFC 1035, 1987.

[14] Squid web proxy cache.
http://www.squid-cache.org.

[15] A. C-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 17th Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.