Performance Aspects Of Distributed Caches Using TTL-based Consistency

Edith Cohen[1], Eran Halperin[2], and Haim Kaplan[2]

[1] AT&T Labs–Research, 180 Park Avenue, Florham Park, NJ 07932, edith@research.att.com
[2] Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel, {heran,haimk}@math.tau.ac.il

Abstract. Web objects are stored and can be requested from numerous servers, including authoritative "origin" servers and caches. Objects can be modified only by their origin servers and weak consistency with cached copies is maintained by limiting their lifetime durations. Copies fetched from origin servers are received with maximum time-to-live (TTL) that equals their lifetime duration whereas copies obtained through a cache have shorter TTLs since their *age* (elapsed time since fetched from the origin) is deducted from their 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. Performance is measured by the number of requests constituting cache misses. 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. More subtly, the number of misses depends on the particular configuration of higher-level caches, e.g., whether one or more higher-level caches are used. Guided by practices for Web caching, we model and compare different configurations. We also analyze the effect of *pre-term refreshes* at high-level caches and *extended lifetimes* at low-level caches and reveal patterns that may seem counter-intuitive at first. Even though TTL-based consistency is very widely used, our work seems to be the first to formally analyze it. Our analysis yields insights and guidelines for improving the performance of Web caches.

1 Introduction

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). Indeed, caching and replication are widely deployed for reducing load on Web servers, network load, and user-perceived latency. Replicating servers are located at different points in the network and include reverse proxies, proxy caches, and browser caches. By and large, Web servers do not maintain per-client state and thus are not aware of all locations of cached copies of objects they serve. Currently the only widely supported and deployed mechanism for consistency of cached copies is client-driven and expiration-based: the authoritative server provides an expiration time for each copy, beyond which it must be validated or discarded. Since this mechanism is the underlying motivation for our model, we further elaborate on it.

The distribution of Web content is governed by HTTP (Hyper-Text Transfer Protocol) [4]. 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 [4, 1, 7] 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 [7, 5]. They are also sometimes configured in hierarchies [6]. 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, 2].

More generally, the miss rate also varies according to the configuration of higher-level caches. We compare different basic cache configurations according to the above notions of age, TTL, and miss rates. In particular, 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 then use extensions of the basic configuration models that incorporate two further cache behaviors [2]. 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 (note that the benefit can be large since a single cache can serve numerous clients). 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.

[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.

Although not typically the case, different caches can use different freshness lifetime durations for the same copy. [2]. Caches that use longer lifetime durations relative to their source 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.

We also consider worst-case behavior and show that a cache which forwards request to another cache (even one that performs pre-term refreshes) can incur twice the miss-rate of a cache that uses an authoritative source. This worst-case ratio can approach e if the cache forwards requests to different caches.

In Section 2 we present the basic and extended models. In Section 3 we provide general results for the basic models, including comparison of different cache configurations. Section 4 provides general results on pre-term refreshes and freshness-lifetime extensions. Section 5 includes worst-case analysis that provides tight bounds on the competitive ratio of different configurations. We conclude in Section 6.

2 Model

We 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 the copy is served. Our analysis focuses on lifetime values that are fixed throughout time. That is, the origin always provides lifetime of T. We differentiate between *origin* (authoritative) servers and *caches* (replicating servers). 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*. 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 (see

[2] The majority of HTTP requests involve objects without explicit expiration and for which the freshness lifetime is determined by a heuristic. Thus, different caches can associate different freshness lifetime values with the same copy as a result of different URL filters and selecting different values for CONF_PERCENT or CONF_MAX [7].

Figure 1) 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 Source Types

In [2] we defined three types of *sources* which capture different relationships between a client-cache and its data source(s). The different sources are illustrated in Figure 1. The TTL value obtained through each source

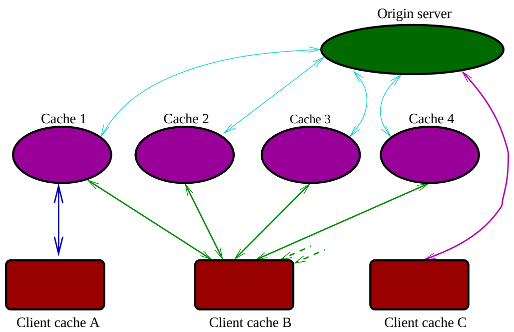

Fig. 1. Different types of sources: Cache A uses cache 1 as an EXC source. Cache B uses caches 1,2,... as an IND source. Cache C uses an AUTH source.

as a function of time is illustrated in Figure 2.

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

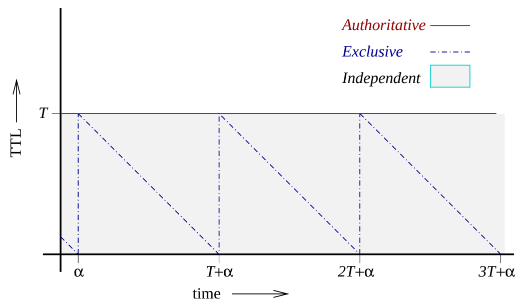

Fig. 2. TTL obtained through different types of sources. AUTH source always provides zero age and TTL of T. EXC source (shown with displacement α) provides age that cycles from 0 to T and thus TTL which cycles from T to 0. IND source provides at each point in time age drawn independently from $U[0,T]$ and thus TTL drawn from $U[0,T]$.

call α the *displacement*. At time t an EXC source provides a copy whose age is $(t-\alpha)$ mod[3]T (TTL equals to $T-(t-\alpha)$ mod T). Note that 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.

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

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).

Two properties used later on are *consistency of a source* and *synchronization* of the relation between a cache and its source. 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.

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.

2.2 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-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.

2.3 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*.

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 forward the request to a different independent EXC_v source. Hence, IND_v source returns copies with age drawn from $U[vT, T]$ and thus TTL drawn from $U[(1 - v)T, T]$. 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, $\text{EXC}_0 \equiv$ AUTH and $\text{IND}_0 \equiv$ 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 Relations Between the Basic Sources

In this section we demonstrate some basic relations between the performance 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 working through an AUTH is never greater than through an EXC, which is in turn at most the miss rate through IND. To do 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 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 not smaller than the TTL available from s_2. Then, for any sequence of requests, the number of misses through s_1 is no larger than 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 length 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.

The proof of the following lemma is given in the full version of the paper

Lemma 2. *Consider two sources s_1 and s_2 that serve objects with TTLs that are independently drawn from distributions with CDFs (Cumulative Distribution Functions) F_1 and F_2, respectively. Suppose that for all $x \geq 0$, $F_1(x) \leq F_2(x)$ (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.*

An immediate consequence of Lemma 1 is that AUTH has smaller miss rate than other source types. The following Lemma establishes that EXC always outperforms IND.

Lemma 3. *For any sequence of requests, the miss rate through an* EXC *source is never higher than through an* IND.

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

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

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 shows that on average over all displacements, EXC performs at least as well.

4 Rejuvenations and Extensions

In this section we show that there is a relation between rejuvenating sources (EXC_v sources) and clients extending the freshness lifetime of the object ($\text{EXC}(r)$ sources). We state the following theorem (its proof will be given in the full version of the paper):

Theorem 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.*

Notice that Theorem 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. *Consider two client caches* c_1 *and* c_2 *that both use either an* AUTH*,* EXC*, or an* IND *sources. Consider a request sequence. If either of the following holds*

- c_1 *uses the same lifetime value as the source.* c_2*, upon some cache misses, uses an extended lifetime.*
- *The object has lifetime of* T*.* c_1 *uses lifetime of* $r_1 * T$ *and* c_2 *uses lifetime of* $r_2 * T$ $(r_2 \geq r_1 \geq 1)$*.*

Then the miss-rate of c_2 *is at most that of* c_1*.*

Proof. The proof for AUTH follows from Lemma 1. The proof for EXC with respect to any fixed displacement follows from Lemma 1, and still holds when we take the expectation over all displacements. The claim for IND source follows from Lemma 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 5. *Consider an* 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* EXC$(\lfloor r \rfloor)$*,*

Proof. When the object expires at the subsidiary, the copy at the parent has age $(r - \lfloor r \rfloor)T$. The object is requested 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. As a corollary of Theorem 1 and Lemma 5, we get the following lemma:

Lemma 6. *Let* $v < 1$*, and let* $u = \frac{1}{\lfloor 1/v \rfloor}$*. 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 1, m_v equals the miss rate through EXC(r) on $\{rt_i\}$. 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 5 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 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 $\{\lfloor r \rfloor t_i\}$ we have a miss in the first T-interval after the extended lifetime of the last miss is over. Hence, we have a miss every $\lfloor r \rfloor T$ time units, and thus, the claim holds.

Lemma 6 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 6 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, IND sources do exhibit monotonic dependence of the miss rate on v:

Lemma 7. *On any sequence of requests through an* IND_v*, the miss rate is non-increasing with* v.

The lemma follows from the following corollary of Lemma 2.

Corollary 1. *Let* $m1$ *be the expected miss rate at a client cache when it uses* IND *source. Suppose now that servers in the source cluster use rejuvenation and let* $m2$ *be the respective expected miss rate. We assume that rejuvenations are independent of the request sequence. That is, the TTL distribution through a uniformly sampled server is fixed over time. Then* $m1 \geq m2$.

Proof. The TTL distribution is $U[0, T]$ without rejuvenation. With rejuvenation, the TTL upon a miss is independently drawn from some other distribution $R[0, T]$. Clearly the CDFs of R and U satisfy that for every t, $F_R(t) \leq F_U(t)$ and Lemma 2 applies.

The following corollary of Lemma 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 2. *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 follows by Lemma 1.

In particular, 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 the full version of the paper 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 supremum over all sequences $\{t_i\}$ of $\rho(\{t_i\}, S)$ is called the competitive ratio of S, and is denoted by $\rho(S)$. Let P be a consistent source that may decide occasionally to rejuvinate the object based on requests that it obtained so far. We call such a source a *rejuvinating policy*. We first analyze the miss rate of rejuvinating policies. Notice that for any given v, EXC_v is a rejuvinating policy (that simply takes decisions which are independent of the sequence). We next show that the competitive ratio of any rejuvinating policy is no greater than 2.

Theorem 2. *The competitive ratio of any rejuvinating 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. Since the rejuvinating policy is consistent the sum of the TTL obtained at the second miss, and the elapsed time between the two misses must be at least T. Thus, a subsequent miss can occur only after at least T time units past the first miss. We now show that the miss rate of every policy is at most twice the optimum. 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. In each of these intervals, the AUTH source has exactly one miss, while the rejuvinating policy could have had at most 2, and thus the theorem holds.

Next we establish a corresponding lower bound of 2 on the performance of any rejuvinating policy that cannot rejuvinate the object continously but must have a small gap of ϵ between consecutive rejuvinations.

Theorem 3. *Consider a randomized rejuvenating policy P on a consistent source such that the time interval between any two rejuvenation points must exceed some $\epsilon > 0$. Then P has competitive ratio $\rho(P) = 2$.*

Proof. By Lemma 1 we get that $\rho(P) \leq 2$. Let $\delta > 0$, and we prove that $\rho(P) \geq 2 - \delta$. We use Yao's lemma [8], and construct a distribution on sequences of requests $r_1, r_2, \ldots$ at times $t_1, t_2, \ldots$, such that the miss rate of the optimal policy is always $\frac{1}{2}$, while the expected miss rate of P is $1 - \delta/2$.

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 $U[t_{2k} + 2T, t_{2k} + 2T + \epsilon)$. Using the property of P, in which the smallest distance between any two rejuvenation points is ϵ, it is easy to see that for every $k \geq 1$, the probability over the request times distribution, that the TTL of r_{2k-1} when using P is at least $T - \epsilon\delta$, is at most δ. Moreover, r_{2k-1} constitutes a miss under any policy, including P. The request r_{2k} is a miss of P with probability at least $1 - \delta$. Thus, the expected miss rate of P is at least $(2 - \delta)/2$. The optimal policy can rejuvenate just before the client incurs a miss, hence has a miss rate of $1/2$. Therefore, using Yao's lemma [8], $\rho(P) \geq 2 - \delta$, and the theorem holds.

In particular, Theorem 2 implies that the source EXC_v with any $v > 0$ has a competitive ratio 2. A related problem is the competitive ratio of IND_v. Since an IND_v source is not consistent it is not a rejuvinating policy and therefore Theorem 3 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 Theorem 2 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 4. *The competitive ratio of* IND_v *is* $\sum_{k=0}^{\lfloor \frac{1}{1-v} \rfloor} \frac{(1-k(1-v))^k}{k! v^k}$.

Thus, by Theorem 4, for $v < 1/2$, the competitive ratio of an IND_v source is 2, and as v approaches 1, the competitive ratio approaches $e = 2.718...$. Interestingly, this worst-case analysis supports the relation between basic sources established in Section 3: EXC source has competitive ratio of 2 whereas IND source has a strictly worse competitive ratio of e. The proof of Theorem 4 is provided in the full version and is based on establishing that the competitive ratio of IND_v equals the expectation of Y_v, where Y_v is defined as follows. Let $X_1, X_2, \ldots$ be a sequence of independent random variables drawn from $U[T(1-v), T]$. Define Y_v as the random variable, such that $Y_v = k$ if and only if $X_1 + \ldots + X_{k-1} \leq T < X_1 + \ldots + X_k$.

6 Conclusion and Open Problems

We modeled and analyzed 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 Pareto arrivals and traces from Web content caches and analysis of request sequences with Poisson and fixed inter-request-time distributions (see [2] and the full version). In particular, these results further demonstrate the presence of interesting dependencies of performance on the extension factor or rejuvenation interval values. Interestingly, the patterns 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.

References

1. T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol — HTTP/1.0. RFC 1945, MIT/LCS, May 1996.
2. E. Cohen and H. Kaplan. Aging through cascaded caches: performance issues in the distribution of Web content. In *Proc. of ACM SIGCOMM*, 2001. To appear.
3. E. Cohen and H. Kaplan. The age penalty and its effect on cache performance. In *Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems*, 2001.
4. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, and T. Leach, P. Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. RFC 2616, ISI, June 1999.
5. Inktomi Traffic Server. `http://www.inktomi.com`.
6. A Distributed Testbed for National Information Provisioning. `www.ircache.net`.
7. Squid internet object cache. `http://squid.nlanr.net/Squid`.
8. A. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 17th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.