Cacheable Web Objects: Understanding Their Modification Times

Matthias Vallentin

vallentin@cs.berkeley.edu

December 9, 2010

Caching forms an integral part of the Internet to avoid duplicate object transfers and save bandwidth. However, little research went into the study of caching from the network perspective. We explore this path by studying the modification process of web objects which can be observed indirectly via meta data in downloads. Our model based on renewal processes captures these interactions and we derive the maximum likelihood estimator. To test our model, we group similar objects using a mixture model. Moreover, we analyze the modification process of individual objects using spectral analysis of the lifetimes and compare different object spectra. This project is a first step in tackling a specific aspect of the generally complex and messy nature of network traffic data and thus requires further efforts to achieve profound results.

1. Introduction

The ongoing trend towards faster, richer, and more sophisticated web applications makes caching an indispensable component of fulfilling the user's expectations for responsiveness. The purpose of browser cache is to store web content for performance reasons in order to both reduce the server load and avoid unnecessary requests for an unchanged object. To this end, the HTTP protocol defines a variety of caching parameters [4] that control object expiration and validation.

In this paper we study object caching from a network perspective. Previous research investigated caching from different vantage points, in particular caches at the endpoints and intermediate points in the network. For example, Mogul et al. developed a scheme to detect duplicate HTTP message bodies [6], Barford et al. studied the effect of request distribution patterns and their caching implications [2], and Doyle et al. explored the effect of ubiquitous caching on request patterns [3]. However, our approach is different in that it is completely *passive*, i.e., solely based on observing network traffic. To the best of our knowledge, this approach has not yet been explored in the past.

Analyzing caching from the network perspective has both advantages and disadvantages. On the one hand, monitoring network of a large site traffic at a central location yields a cross-section of a various different objects that are being transferred. On the other hand, when an object can be retrieved from a cache rather than from its original source (i.e., a *cache hit*) downstream of the vantage point, it will not show up in our data. Consequently, our observations are confined to *cache misses*. This is a study of rare events: cacheable objects do not show up frequently in the traffic but these are the ones we want to study.

More concretely, this research sets out to infer the modification process of cacheable HTTP objects, whose times of modification are indirectly available as meta data in each object download. We develop a stochastic model to describe this point process and try to understand our data by carrying out both time and frequency domain analysis methods.

The paper is structured as follows: after familiarizing the reader with the relevant aspects of HTTP caching in §2, we introduce our data in §3. Next we describe our model in §4 and link it to the data in §5. Finally, we give concluding remarks and ideas for future work in §6.

2. HTTP caching

Caching in HTTP [4] is based on two principal mechanisms, *expiration* and *validation*. To allow clients to verify the freshness of an object, the server specifies its expiration, using either the

Environment	Start	End	Length	HTTP connections
LBNL	4/21/2010 12:05pm	4/23/2010 12:34pm	48 hours	22,200,000
AirJaldi	10/18/2010 5:16pm	11/9/2010 10:30am	21 days	30,200,000

Table 1: Summary of the two data sets from the Lawrence Berkeley National Laboratory and AirJaldi.

`Expires` header[1] or the `max-age` directive in the `Cache-Control` header. A fresh object at the client does not need to be refetched, thereby avoiding unnecessary requests entirely. An expired object must be validated by asking the origin server (i.e., not an intermediate cache) if the local copy can still be used. The `Expires` header contains an absolute date time stamp, but when we refer to *expiration time*, we mean the absolute expiration value minus the time when the object was downloaded.

3. Data

This section introduces our data sets. In §3.1 we present the two environments from where we obtained the data and §3.2 we describe the process of how we filter, clean, and extract the relevant pieces of information of the data.

3.1. Environments

Our data set consists of processed network traffic using the Bro network intrusion detection system [7] from contrasting environments: a large research institute in California, USA, and a rural community in the Indian Himalayas. The high-level details of our data are summarized in Table 1.

The first environment is the Lawrence Berkeley National Laboratory (LBNL, [5]), which is the oldest of the U.S. Department of Energy's national laboratories and managed by the University of California, Berkeley. Its approximately 4,000 users and 13,000 hosts are connected to the Internet via a 10 Gbps uplink. We captured a 48-hour packet trace with a HTTP filter from April 21 to

[1]The HTTP protocols splits requests and responses in header (meta data) and body (data).

April 23 with 22.2 million connections.

The second environment is the AirJaldi [1] rural wireless network serving mostly the Tibetan community-in-exile. The network comprises about 8,000 users and has an uplink totaling 10 Mbps. We processed HTTP traffic for 21 days from October 18 to November 9 with 30 million connections.

3.2. Inspection

We restrict our analysis to HTTP objects that can contain executable code, which are mainly HTML and JavaScript. These object types are particularly interesting because they contain the structure and logic of most of web sites. Furthermore, this study is about understanding objects that are cacheable and henceforth we solely focus on those. In both environments, images account for the largest share of cacheable objects, cacheable HTML and JavaScript only account for 5.27 / 2.94 % and 4.06 / 3.67 % respectively for LBNL/AirJaldi.

4. Model

In this section we develop a stochastic model that formalizes interactions between downloads and object modification times. In §4.1 we present our point process model, followed by a maximum likelihood analysis in §4.2 and a simple instantiation in §4.3

4.1. Stochastic Process

We model object lifetimes as a *renewal process* $\{L_t : t \in T\}$, i.e., a set of independent, identically distributed (IID) positive random variables (RV's) each of which represents the *lifetime* of an object until its next modification. $T \subseteq \mathbb{N}$ is an index set to label the RV's. The random times at which a modification occurs form the point process $\{M_t : t \in T\}$. Hence, their pairwise differences $L_t \triangleq M_{t+1} - M_t$ form the lifetime of a particular object version. This process ranges from the birth of an object at time M_0 until its destruction. An illustration is visualizes these processes in

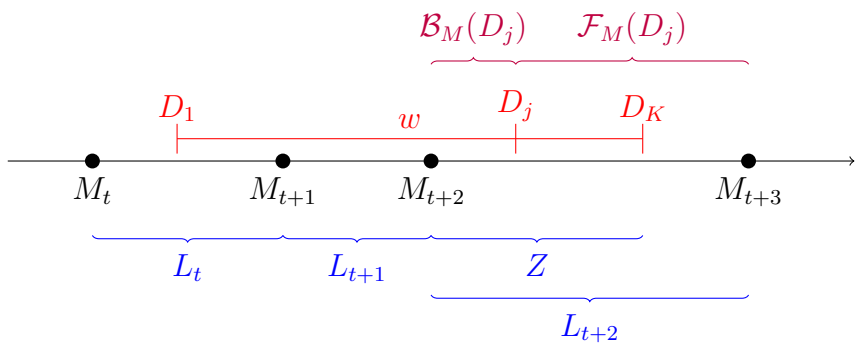


Figure 1: An illustration of our model. Object *download times* are denoted by D_j where $1 \leq j \leq K$ and their *modification times* by M_t. Each object version t has a *lifetime* L_t. During the observed modifications in the measurement window w, there exists at least one censored lifetime L_{t+n} which we denote by Z. The *backward recurrence time* $\mathcal{B}_M(D_j)$ is the time elapsed since the last modification at the time of download j and the *forward recurrence time* $\mathcal{F}_M(D_j)$ the time to the next modification of M.

Figure 1.

Downloads in our data sets correspond to the times where we can observe the otherwise hidden process M_t. This is because each download includes the `Last-Modified` header that reveals the last M_t value. Technically, the downloads also form a stochastic process $\{D_j : j \in \mathbb{N}\}$, but we consider them as fixed for now. In future work plan to extend this model to *double stochastic* processes.

Let $w = D_K - D_1$ denote the discrete measurement window with K downloads of the same object in which we observe $n-1$ object modifications with corresponding lifetimes L_{t+n-1}, where $K > n$. The last modification lies outside w and the corresponding lifetime L_{t+n} is thus always *censored*, i.e., truncated at D_K.

The *backward recurrence time* and *forward recurrence time*

$$\mathcal{B}_M(s) \triangleq s - M_t \qquad M_t \leq s < M_{t+1}$$

$$\mathcal{F}_M(s) \triangleq M_{t+1} - s \qquad M_t < s \leq M_{t+1}$$

relate M_t and D_j. The index M of $\mathcal{B}$ and $\mathcal{F}$ denotes the process being referred to. The last lifetime

for $n-1$ observations of M_t can now be written as $L_{n-1} = \mathcal{B}_M(D_K) + \mathcal{F}_M(D_K)$. We observe zero or more L_t samples and exactly one $\mathcal{B}_M(D_K)$ sample per object. For notational convenience, we define $Z \triangleq \mathcal{B}_M(D_K)$.

Another way to think of this model is that the first download immediately yields an observation $Z = \mathcal{B}_M(D_1)$ and subsequent downloads merely update Z if no modifications occurred in the meantime. That is, if j downloads occur until the next modification M_{t+1}, Z is being redefined as $\max_j Z = \mathcal{B}_M(D_j)$. As soon as a new download D_{j+1} reveals M_{t+1}, we have a L_t sample and reset $Z = 0$.

4.2. Maximum Likelihood Estimation

We now turn to the problem of parameter estimation by deriving the likelihood function. Before we instantiate the model with particular distribution functions, we derive the likelihood in a general form, similar to the approach taken by Zhao [8]. For a given object, let $L_t \overset{iid}{\sim} F$ with PDF f and let θ be the parameter vector of F. We denote observed data with lowercase letters, e.g., l_t is an observed lifetime sample of L_t. Then, the likelihood of θ is

$$\mathcal{L}(\theta) = \prod_{i=1}^{n-1} f(l_i) \int_z^\infty f(t)\, dt = \prod_{i=1}^{n-1} f(l_i)(1 - F(z))$$

The first factor of the product represents the likelihood of all L_t samples. The second factor is the density of the Z sample which has to be integrated over the remaining unknown space due to censoring. When $n = 1$, we have only a single observation z and the empty product of densities on the right side evaluates to one.

Moreover, for N IID objects we can group together the number of L_t and Z samples. Let N_L and N_Z be the total number of L_t and Z observations. Then, the joint likelihood is

$$\mathcal{L}_N(\theta) = \prod_{i=1}^{N_L} f(l_i) \prod_{i=1}^{N_Z} (1 - F(z_i)).$$

Unit	seconds	minutes	hours	days	weeks	months	years
Letter	s	m	h	d	W	M	Y

Table 2: Unit shortcuts used on the time axes of the plots.

4.3. Model Instantiation

A basic instantiation of this model associates a homogeneous Poisson process with each object X having parameter λ where $M_t \sim \text{Gamma}\,(t, \lambda^{-1})$ and $L_t \sim \text{Exp}\,(\beta)$ where $\beta = \lambda^{-1}$. Further, N IID objects $(X_1, \ldots, X_N)$ have the same parameter λ.

Theorem 1. *The maximum likelihood estimator (MLE) for N censored IID observations with $L_t \overset{iid}{\sim} Exp\,(\beta)$ has a closed form and is given by*

$$\widehat{\beta} = \begin{cases} \dfrac{\sum_{i=1}^{N_L} l_i + \sum_{i=1}^{N_z} z_i}{N_L} & N_L > 0 \\ \max\limits_i z_i & N_L = 0 \end{cases}.$$

The proof can be found in Appendix A. Other instantiations with L_t being distributed as Gamma, Weibull, or Lognormal, require numerical methods to find the MLE. We will continue the discussion of this simple model in the following section.

5. Analysis

Having a stochastic model in place, we turn in §5.1 to the analysis of the data with the goal to validate our model. In §5.2 we then perform a more fine-grained object examination.

5.1. Clustering by Expiration Times

Ultimately we are interested in parameter estimates of θ. Although some objects could stem from the same distribution, it is quite unlikely that every object in the entire data set follows a process with the same parameter. Therefore, clustering according to some observable feature is desirable.

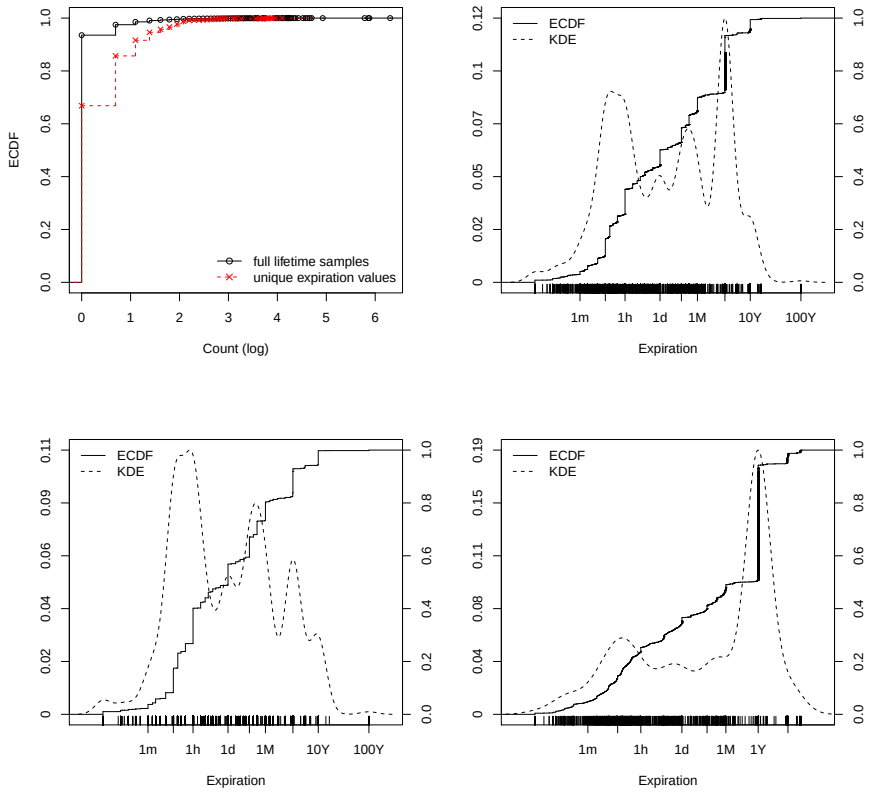


Figure 2: Analysis of expiration values of the JavaScript objects in the AirJaldi data set. The top left panel shows the ECDFs of *(i)* the number of objects that have more than L_t sample and *(ii)* the number of unique expiration values. The top right panel shows the expiration distribution of objects for which *(i)* holds, and the lower two panels display the distribution for constant and varying expiration values.

The object expiration time is an interesting candidate for clustering because it should represent a lower bound of the actual modification times[2] Intuitively, we thought that the expiration would be fix and does not change over time, i.e., that it represents a constant offset from the time the object is being downloaded. When looking at the objects with at least one full uncensored L_t sample, which are 6.5/3.7 % for JavaScript/HTML in the AirJaldi data set, we find that only 66.8/72 % of the objects have a constant expiration. We continue to use the AirJaldi data set because it spans a full month compared to the 2-day LBNL data set (see §3.1). The top left panel of Figure 2

[2]Recall that machines will keep using their local object in the cache until it expires.

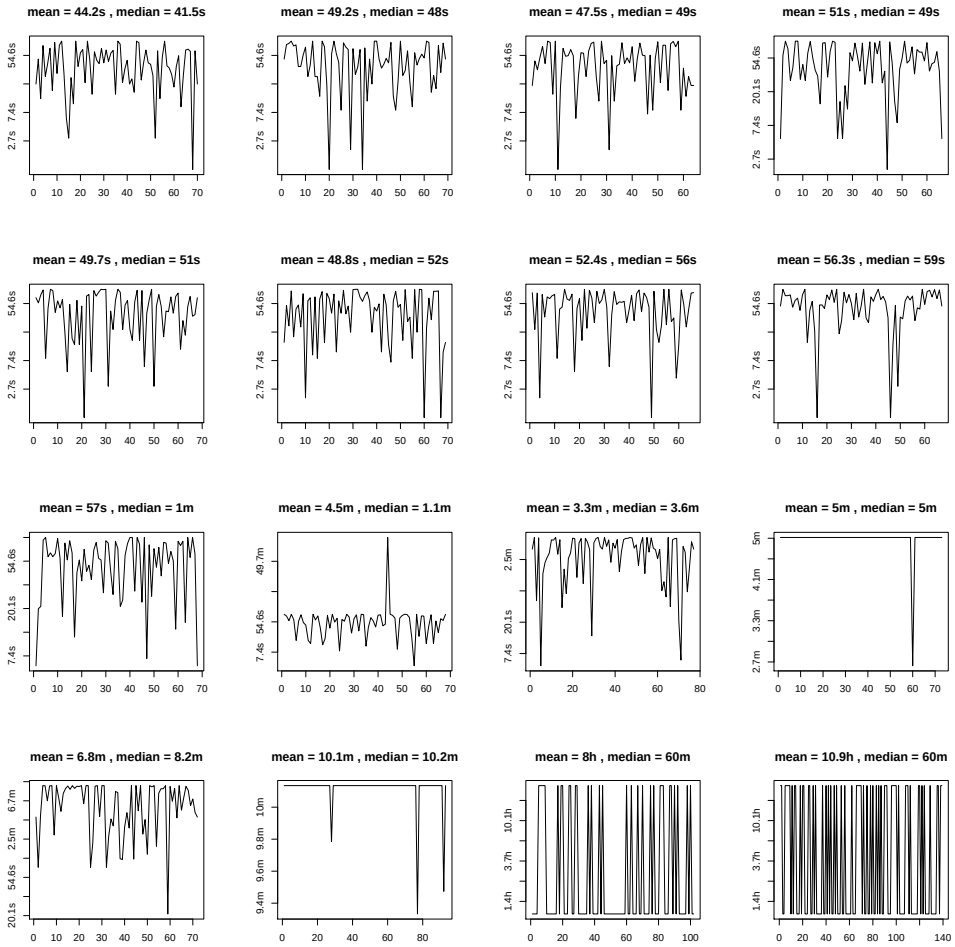


Figure 3: The changing values of expiration times for the 16 objects having the largest number of L_t samples. The x-axis shows the index t of the L_t samples and is thus not linear in time. The y-axis displays the value of the expiration time on a logarithmic scale.

shows the ECDF for number of samples with at least one L_t sample. Then, for objects having this property, the same plot also includes a dashed ECDF of the number of unique expiration values observed. Around 30 % of the objects have varying expiration values. The top right panel shows the distribution of the expiration values of all objects, the lower left panel only of objects with constant expiration, and the lower right panel only of objects with varying expiration.

We revise our earlier intuition and now believe that two types of expiration models exist: *(i)* an object with constant expiration time that represents a fixed offset into the future and *(ii)* an

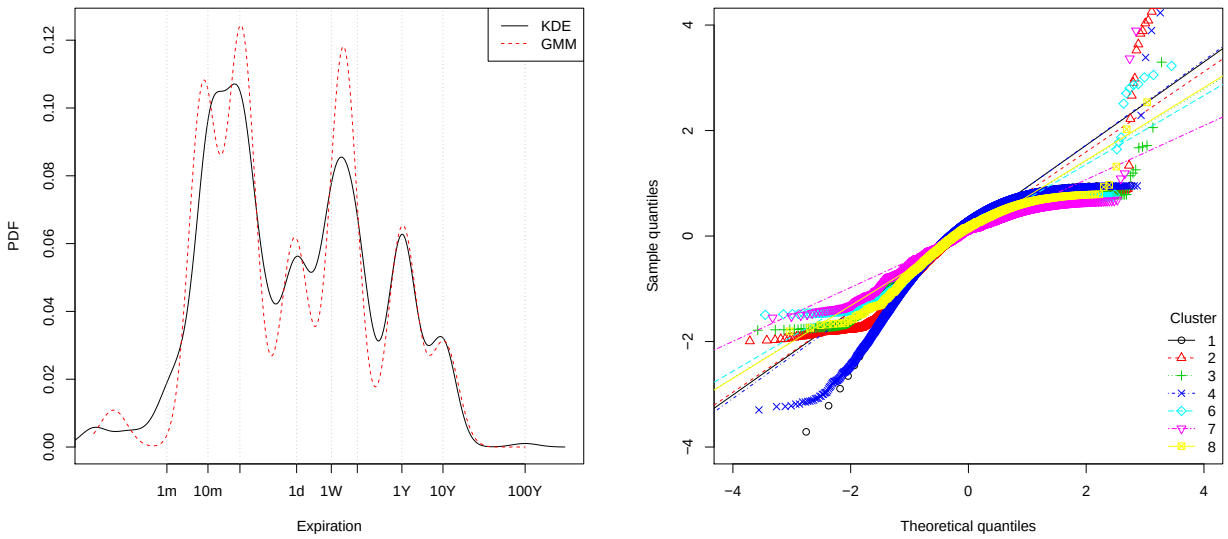
object with absolute expiration value of a fixed point in time in the future. Objects adhering to the second model would exhibit expiration times that monotonically decrease until the expiration point is reached. To test our hypothesis, we plot in Figure 3 the expiration times of the 16 objects with the largest number of L_t samples. The x-axis shows the index t of the L_t samples and is thus not linear in time. The y-axis displays the value of the expiration time on a logarithmic scale. The time axis labels are explained in Table 2. We observe that many objects (panels 1-9, left to right) with mean expiration values close to one minute fluctuate, which could be indicative for model *(ii)* where downloads hit uniformly in this interval. For the first three panels of the first two rows, this could be the case: the Kolmogorov Smirnov goodness-of-fit test returns p-values greater than 0.1.[3]

For the following analysis, we discard objects with varying expiration times and cluster the remaining objects accordingly. That means that we only consider the object with an expiration time distribution of the lower left panel in Figure 2. Each stark discontinuity in the ECDF correspond to a mode. Note that the modes align with easy to remember time values, such as one hour, one day, or one year.

Multimodal densities often reflect the existence of subpopulations, which motivates the use of a *mixture model* to cluster the objects according to their expiration time. We employ a Gaussian mixture model (GMM), i.e., a convex combination of G normally distributed *mixture components* with PDF $\phi(x; \mu_i, \sigma_i^2)$. The components are weighted by *mixing proportions* that are given by a latent multinomial random variable Y with parameter π, where $\pi_i = p(Y^i = 1 \mid \pi)$ are constrained to sum to one. The univariate mixture model is given by $p(x \mid \theta) = \sum_{i=1}^{G} \pi_i \phi(x; \mu_i, \sigma_i^2)$. Figure 4a shows how this GMM approximates the subpopulations compared to a non-parametric kernel density estimator (KDE). Here, we compare the KDE to the mixture density obtained via the EM algorithm for JavaScript expiration times from the LBNL data set.

The next step is to analyze each component in isolation. We test whether the L_t samples come

[3]Unfortunately we cannot trace back the object because we have only high-level aggregate data in this data set. For the next measurement we will include the necessary information to validate this assumption.



(a) Comparison of estimated expiration time densities. A non-parametric kernel density estimator (KDE) is compared to a Gaussian mixture model (GMM) with 7 components.

(b) Testing for exponential lifetime distribution after clustering objects according to the classification from GMM shown in the left panel.

Figure 4: Empirical analysis of object expiration times and fitting of a mixture model.

from an exponential distribution. If this is the case, we can continue with maximum likelihood estimation per cluster using the analytical solution found in §4.3. As shown in Figure 4b, the per-cluster sample quantiles do not lie on a straight line, indicating that the exponential assumption is not a good starting point. We plan to investigate different model instantiations in the future that are a better fit for this type of data.

5.2. Point Process Examination

While the previous section examined expiration times as possible means to group objects together and then conduct per cluster analysis, we now look at the individual object level and seek to understand the behavior of the point process M_t in detail. As before, we restrict our analysis to JavaScript objects from the AirJaldi data set.

Figure 5 shows the first-order differences $l_t = \nabla m_t$ of the 9 processes having the largest number

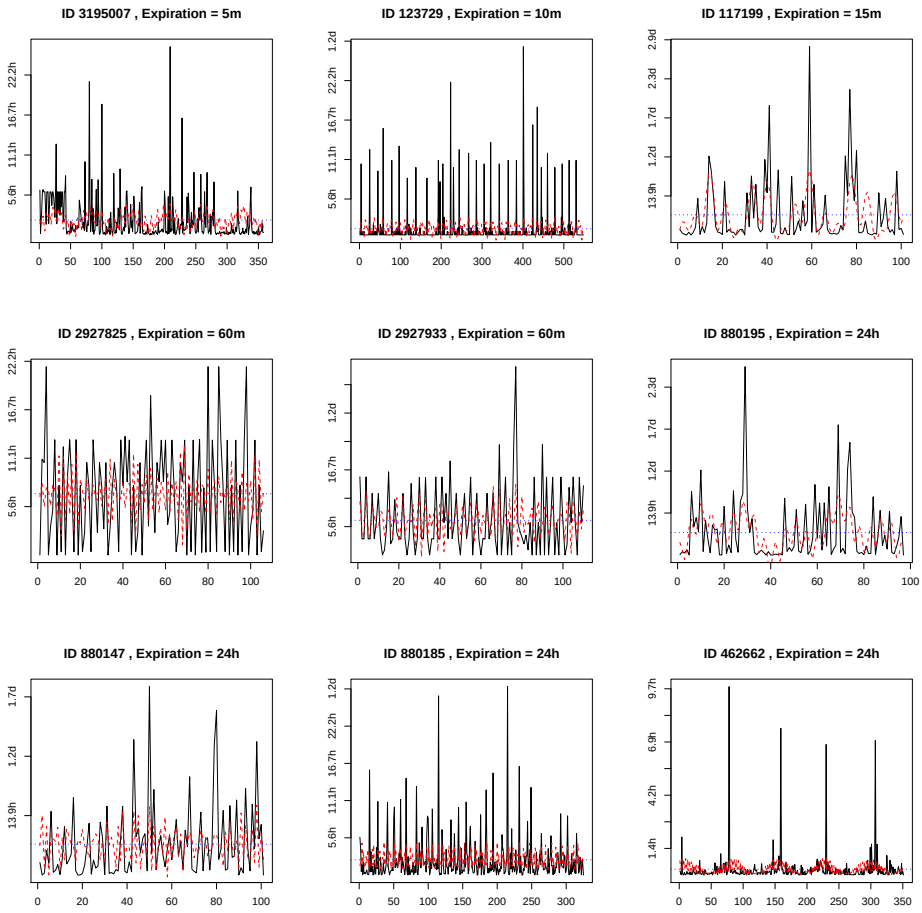
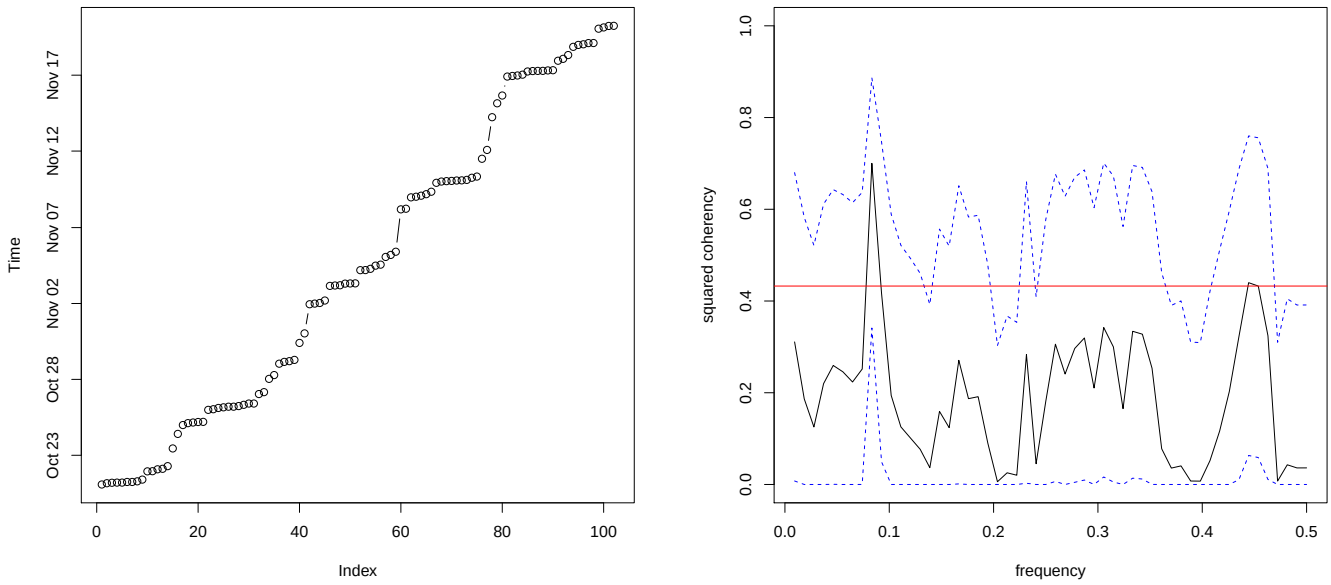


Figure 5: Differences of the residual process of object modification times. The red dashed line is a periodic fit with the three most dominant frequencies from the periodogram. The panels are sorted by expiration time and each one shows a different object (with a unique ID).

of samples. Note that applying the difference operator ∇ renders the process stationary in its index. The x-axis shows the event number and the y-axis time. We also fit a periodic model of the form

$$\widehat{l_t} = a_0 + \sum_{j=1}^{q} [a_j \cos(2\pi\omega_j t) + b_j \sin(2\pi\omega_j t)]. \tag{1}$$

From the scaled periodogram of l_t we extract the $q = 3$ most dominant frequencies ω_j. The red dashed line visualizes this fit and the dotted blue line shows the intercept $a_0 = \bar{x}$. After manually doing this process for a few objects we wrote R functions to automate this procedure; the code



(a) Residual process $\{m_t\}$ of object 117199 (top-right).

(b) Squared coherency of l_t of object 117199 (top-right) and 880147 (bottom-left).

Figure 6: Examination of one specific residual process $\{m_t\}$ and coherency analysis between two $\{l_t\}$ processes from Figure 5.

can be found in Appendix B.

The large lifetime spikes in Figure 5 do not necessarily mean that the lifetime has in fact that value, rather, we hypothesize that they represents measurement outages introduced by the lack of downloads of a given object. The fit can be regarded as the baseline and the spikes help us to understand when outages occur.

Let us analyze one of the processes in more detail. A plot of the residual process $\{m_t\}$ of object 117199 in the top-right panel is shown in Figure 6a. An inspection of the jumps reveals that they correspond to night times, confirming the intuition that jumps refer to the lack of samples. Object 462662 in the bottom-right panel of Figure 5 has even sharper jumps happening from 12am to 8am. After removing the spikes of the $\{l_t\}$ process, the lifetimes look exponential and for object 462662 the Kolmogorov Smirnov goodness-of-fit yields a p-value of 0.9, indicating that the corresponding

$\{m_t\}$ process is homogeneous Poisson. This contrasts to our result in §5.1 and suggests revisiting our clustering method.

Next we compare two similar looking objects: 117199 and 880147 shown in the top-right panel and bottom-left panel of Figure 5. Figure 6b shows the squared coherency over a band of $L = 10$, $df \approx 13$ and $F_{2,df-2}(.001) \approx 13.72$ at the significance level of $\alpha = .001$. Thus we can reject the null hypothesis of $\rho^2_{y\cdot x}(\omega) = 0$ for values where $\bar{\rho}^2_{y\cdot x}(\omega) > C_{0.001}$, where $C_{0.001} = 0.43$ is depicted by the horizontal red line. The dashed blue lines show the confidence band, which is valid for $\rho^2_{y\cdot x}(\omega) > 0$. The objects appear to be coherent.

6. Conclusion

In this project we set out to understand the modification process of web objects by passively watching network traffic. This process is only indirectly observable via download meta data.

We develop a model based on renewal processes and derive the likelihood function. To exploit the similarity of objects, we cluster them using a Gaussian mixture model and test our model against each cluster. We find that our Poisson process assumption does not hold when looking at the data on a per-cluster basis.

Further, we try to understand the point process of modification times on a per-object basis. By carrying out a spectral analysis of the lifetimes, we find the most dominant frequencies that describe well the modifications despite measurement outages. Further, a comparison of different object spectra confirms that several objects have a similar modification behavior, which encourages the use of clustering techniques.

The presented analysis and model descriptions are still in a very fledgling stage and need further thought to capture the domain-specific properties of web objects. In future work, we plan to try other directions that go beyond renewal processes, such as hidden Markov models. Also, currently we consider the process of object downloads as fixed to simplify the analysis. Promising seems to be the use of a *doubly stochastic point process* of two interwoven processes to express that D_t

yields an observations of M_t. Moreover, we can improve on clustering objects by including other meta data than just the expiration time.

References

[1] AirJaldi. `http://www.airjaldi.org`.

[2] P. Barford, A. Bestavros, A. Bradley, and M. Crovella. Changes in Web client access patterns: Characteristics and caching implications. *World Wide Web*, 2(1-2):15–28, 1999.

[3] R. P. Doyle, J. S. Chase, S. Gadde, and A. Vahdat. The Trickle-Down Effect: Web Caching and Server Request Distribution. *Computer Communications*, 25(4):345–356, 2002.

[4] HTTP/1.1: Caching in HTTP. `http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html`.

[5] Lawrence Berkeley National Laboratory. `http://www.lbl.gov`.

[6] J. C. Mogul, Y. M. Chan, and T. Kelly. Design, Implementation, and Evaluation of Duplicate Transfer Detection in HTTP. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pages 4–4, Berkeley, CA, USA, 2004. USENIX Association.

[7] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23–24):2435–2463, 1999.

[8] Y. Zhao. Parametric Inference from Window Censored Renewal Process Data. Ph.D. Thesis, The Ohio State University, 2006.

A. Proof of Theorem 1

The log-likelihood of $\mathcal{L}_N(\theta)$ is

$$\ell_N(\theta) = \sum_{i=1}^{N_L} \log f(l_i) + \sum_{i=1}^{N_Z} \log(1 - F(z_i))$$

For $L_t \sim \text{Exp}(\beta)$, ℓ_N becomes

$$\begin{aligned}
\ell_N(\theta) &= \sum_{i=1}^{N_L} \log\left(\frac{1}{\beta}\exp\left\{-\frac{l_i}{\beta}\right\}\right) + \sum_{i=1}^{N_Z} \log\left(1 - \left(1 - \exp\left\{-\frac{z_i}{\beta}\right\}\right)\right) \\
&= N_l \log\frac{1}{\beta} + \sum_{i=1}^{N_L} \log\exp\left\{-\frac{l_i}{\beta}\right\} + \sum_{i=1}^{N_Z} \log\exp\left\{-\frac{z_i}{\beta}\right\} \\
&= -N_l \log\beta - \frac{1}{\beta}\sum_{i=1}^{N_L} l_i + -\frac{1}{\beta}\sum_{i=1}^{N_Z} z_i \\
&= -N_l \log\beta - \frac{1}{\beta}\left(\sum_{i=1}^{N_L} l_i + \sum_{i=1}^{N_Z} z_i\right).
\end{aligned}$$

Setting

$$\frac{\partial}{\partial\beta}\ell_N(\beta) = -\frac{N_L}{\beta} + \frac{1}{\beta^2}\left(\sum_{i=1}^{N_L} l_i + \sum_{i=1}^{N_Z} z_i\right) \equiv 0$$

yields

$$\widehat{\beta} = \frac{\sum_{i=1}^{N_L} l_i + \sum_{i=1}^{N_Z} z_i}{N_L} \qquad N_L > 0.$$

Since

$$\frac{\partial^2}{\partial\beta^2}\ell_N(\beta)\bigg|_{\beta=\widehat{\beta}} = -\frac{N_L^3}{c^2} < 0 \qquad \text{where } c = \sum_{i=1}^{N_L} l_i + \sum_{i=1}^{N_Z} z_i$$

we have indeed found the MLE for $N_L > 0$. When $N_L = 0$, the first term in the log-likelihood

vanishes and we need to bound Z to obtain a meaningful result by using an indicator function I:

$$\begin{aligned}\ell_N(\theta) &= \sum_{i=1}^{N_Z} \log\left(1 - \left(1 - \exp\left\{-\frac{z_i}{\beta}\right\}\right)\right) I(0 < z_i < \beta) \\ &= -\frac{1}{\beta}\sum_{i=1}^{N_Z} z_i I(0 < z_i < \beta).\end{aligned}$$

Since this is a strictly increasing function in β, the MLE is at $\widehat{\beta} = \max_i z_i$, resulting in

$$\widehat{\beta} = \begin{cases} \frac{\sum_{i=1}^{N_L} l_i + \sum_{i=1}^{N_z} z_i}{N_L} & N_L > 0 \\ \max_i z_i & N_L = 0 \end{cases}.$$

□

B. Automated Spectral Analysis in R

To facilitate the automated extraction of dominant frequencies and fit a periodic model according to equation 1, we wrote the following R functions.

```
# Extract the top k most dominant frequencies from a periodogram. When
# neighborhood has a value greater than zero, this many neighboring frequencies
# left and right of a dominant frequency are also removed before considering
# the next most dominant one.
# periodogram: a periodogram object, e.g., as returned by spec.pgram().
# k: the number of most dominant frequencies to extract.
# neighborhood: when selecting a dominant function, remove that many neighbors
# around this frequency as well.
top.frequencies = function(periodogram, k=3, neighborhood=0)
{
    spc = periodogram$spec
```

```
    frq = periodogram$freq

    topk = c()
    while (length(topk) < k)
    {
        top = which.max(spc)
        topk[length(topk)+1] = frq[top]

        if (neighborhood > 0)
        {
            idx = seq(top - neighborhood, top + neighborhood)
            spc = spc[-idx]
            frq = frq[-idx]
        } else {
            spc = spc[-top]
            frq = frq[-top]
        }
    }

    topk
}

# Fit a periodic linear model with the given frequencies to a time series.
# Avoiding an intercept term in the fit can be achieved via setting
# intercept=F.
# x: the time series
# f: vector of frequencies to fit.
# intercept: whether to include an intercept term in the regression.
periodic.fit = function(x, frequencies, intercept=T)
{
    idx = 1:length(x)
    sin.fit = function(f) sin(2*pi*idx*f)
```

```
    cos.fit = function(f) cos(2*pi*idx*f)

    f = lapply(frequencies, function(f) cbind(cos.fit(f), sin.fit(f)))
    covariates = sapply(1:length(f), function(i) paste("f[[", i, "]]", sep=""))
    add = function(a, b) paste(a, b, sep= " + ")
    covariates = Reduce(add, covariates)

    form = "x ~"
    if (! intercept)
        form = paste(form, 0, "+")
    form = paste(form, covariates)

    lm(as.formula(form))
}

# Example usage of the above functions.
x1 = 2*cos(2*pi*1:100*6/100) + 3*sin(2*pi*1:100*6/100)
x2 = 4*cos(2*pi*1:100*10/100) + 5*sin(2*pi*1:100*10/100)
x3 = 6*cos(2*pi*1:100*40/100) + 7*sin(2*pi*1:100*40/100)
x = x1 + x2 + x3
spc = spec.pgram(x, taper=0, log="no")
freqs = top.frequencies(spc)
fit = periodic.fit(x, freqs)
plot.ts(x)
lines(fitted(fit), col="red", lty=2)
```