Modeling Object Characteristics of Dynamic Web Content

Weisong Shi, Eli Collins, and Vijay Karamcheti
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
Technical Report TR2001-822
{*weisong, vijayk*}*@cs.nyu.edu*

Abstract

Requests for dynamic and personalized content increasingly dominate current-day Internet traffic, driven both by a growth in dynamic web services and a "trickle-down" effect stemming from the effectiveness of caches and content-distribution networks at serving static content. To efficiently serve this trend, several server-side and cache-side techniques have recently been proposed. Although such techniques, which exploit different forms of reuse at the sub-document level, appear promising, a significant impediment to their widespread deployment is (1) the absence of good models describing characteristics of dynamic web content, and (2) the lack of effective synthetic content generators, which reduce the effort involved in verifying the effectiveness of a proposed solution.

This paper addresses both of these shortcomings. Its primary contribution is a set of models that capture the characteristics of dynamic content both in terms of independent parameters such as the distributions of object sizes and their freshness times, as well as derived parameters such as content reusability across time and linked documents. These models are derived from an analysis of the content from six representative news and e-commerce sites, using both size-based and level-based splitting techniques to infer document objects. A secondary contribution is a Tomcat-based dynamic content emulator, which uses these models to generate ESI-based dynamic content and serve requests for whole document and separate objects. To validate both the models and the design of the content emulator, we compare the bandwidth requirements seen by an idealized cache simulator that is driven by both the real trace and emulated content. Our simulation results verify that the output of the content emulator effectively and efficiently models real content.

Keywords: Object Characteristics, Dynamic Web Content, Dynamic Content Emulator

Wordcount: 7934

1 Introduction

Dynamic and personalized content delivery has attracted a great deal of attention recently from both commercial and research communities. One reason is of course the growing popularity of dynamic web services, exemplified by news sites and personalized sites (e.g., myyahoo.com) both of which require dynamic generation of content. The other reason is the trickle-down effect [15] of widely deployed proxy caches and content delivery networks (CDNs), which effectively filter incoming requests for static web content and presumably shift the dominant loads seen on the Internet from popular static objects to less popular objects and dynamic web content.

To efficiency serve and deliver such dynamic and personalized content, researchers have recently proposed several server-side and cache-side mechanisms. Server-side techniques, exemplified by techniques such as data update propagation [8], fragment-based page generation [9, 14], and the Cachuma [35] class-based page classification

scheme reduce the burden on the server by allowing reuse of previously generated content to serve new requests. Cache-side techniques, exemplified by systems such as Active Cache [7], Gemini [24], CONCA [29], and the content assembly technique proposed by Wills et al [32], attempt to reduce the latency of dynamic content delivery by moving some functionality to the edge of network. Similar trends are also visible in commercial caching and edge server products, most notably IBM's WebSphere [10] and Akamai's Edgesuite [18]. Despite their focus on different aspects of the problem, these approaches share the same rationale, specifically that it is possible to view the document in terms of a quasi-static *template* (expressed using formatting languages such as XSL-FO [16] or edge-side include (ESI) [30]), which is filled out with multiple individually cacheable and/or uncacheable *objects*. This object composition assumption enables surrogates and downstream proxy caches to reuse templates and cached objects to efficiently serve subsequent requests and additionally reduces server load, bandwidth requirements and user-perceived latencies by allowing only the unavailable objects to be fetched.

Although the above techniques appear promising, it is difficult to predict to what extent their stated benefits apply to a workload different than the one they were evaluated on. As an example of the challenges, consider the following questions that we were faced with when trying to come up with general policies for our CONCA architecture [29]: At what granularity must objects be cached to achieve sufficient reuse? Is their a sharp threshold for choosing this granularity, or is it the case that the benefits are continuously varying? Can we estimate likely freshness times of objects from the duration they have been present in the cache? Is their a correlation between object size and their reuse?

While trying to answer these questions we encountered the problem that unlike the wealth of models on the nature of static web content, ranging from the zipf-like request distribution to the pareto distribution of document sizes [22], there is an absence of anything corresponding for dynamic content. Even if such models were present, an additional problem one encounters while trying to evaluate a new technique is the absence of template-based content. At this time, web servers do not explicitly supply the template and component objects making up the document (even if to a human eye, it is often clear what these are). The lack of such content rules out simulation-based studies of the kind that have resulted in tremendous advances in the state of the art for delivery and caching of static content.

This paper addresses both of these shortcomings. Its primary contribution is a set of models that capture the characteristics of dynamic content both in terms of independent paramters such as the distributions of component object sizes and their freshness times, as well as derived parameters such as whole-document content reusability across time and linked documents. To derive these models, we analyzed dynamic content collected every five minutes over a two-day period from six representative news and e-commerce sites: `www.cnn.com`, `dailynews.yahoo.com`, `www.nytimes.com`, `www.amazon.com`, `www.barnesnoble.com`, and `www.windowsmedia.com`. To cope with the absence of an explicit template in the documents, we inferred both the template and the component objects using parameterized level-based and size-based splitting techniques. The main analysis results are summarized below:

- The sizes of component objects making up a dynamic web page can be captured very well using an *Exponential* distribution. This is in contrast to how static documents are modeled, usually with a heavy-tailed pareto distribution.

- Except for a considerable fraction of objects that change very infrequently, the freshness times of component objects can be captured using a *Weibull* distribution. For two of the sites above (`www.amazon.com` and

www.bn.com), this distribution degenerates into a sharp bimodal one: all of the objects either change on almost every access or change very infrequently.

- Content from all of the six sites demonstrated significant opportunity for reuse across both the temporal and spatial (in linked documents) dimensions. More interesting is the relationship between content reuse and the object granularity: for four of the sites (www.cnn.com, dailynews.yahoo.com, www.nytimes.com, www.windowsmedia.com), there was a graceful degradation of reusability with increasing object granularity, while the degradation was much sharper for the other two.

A secondary contribution of the paper is a dynamic content emulator (DCE), which uses these models to generate parameterizable dynamic content that adheres to the ESI specification. DCE builds on top of the Tomcat web server from the Jakarta open source initiative, and can service requests for both whole documents as well as individual components. DCE is easy to configure, extend, and use and should prove a useful tool for other researchers in this area. To validate both our models and their use in DCE, we wrote an idealized cache simulator that can works off both the real trace data as well as the output of DCE. Comparing the outputs of this simulator for the two cases verifies that DCE effectively models real content, and at a significant simulation cost advantage.

The rest of this paper is organized as follows. Section 2 describes the methodology used in this study, specifically the extraction of information about document templates and component objects. The analysis results for content from the six sites and derived models are presented in Section 4. Section 5 describes the design, implementation, and validation of the stand-alone dynamic content emulator. Section 6 discusses related work and we conclude in Section 7.

2 Characterizing Dynamic Content

We first discuss the metrics of interest in dynamic content and then describe our approach for characterizing them.

2.1 Metrics of Interest

The consensus view for thinking about dynamic content appears to be based upon the notion of object composition. A document is viewed as consisting of a quasi-static document *template* and individual *objects* exhibiting different characteristics, which fill out this template. These characteristics can include attributes such as whether the object is cacheable or not, what its update frequency is, whether it can be shared across multiple requests/users or if it is personalized, etc. Figure 1 shows the snapshot of a personalized myyahoo!.com page, and what the corresponding document template and component objects might look like. S* and P* represent objects that are shared and private respectively, and TTL captures the length of time this object remains valid.

From the perspective of surrogates and proxy caches that are attempting to improve delivery of dynamic content, there are several metrics of interest. Clearly both the *number of objects* and their *size distribution* are important: the first indicates the opportunity for reuse, while the second determines whether or not this opportunity can be traded off against object management overheads. Another metric of interest is the document-level *content reusability*, both across repeated requests for the same document (the temporal dimension) as well as across requests for other documents that are linked from this one (the spatial dimension). Content reusability, which we define to be the

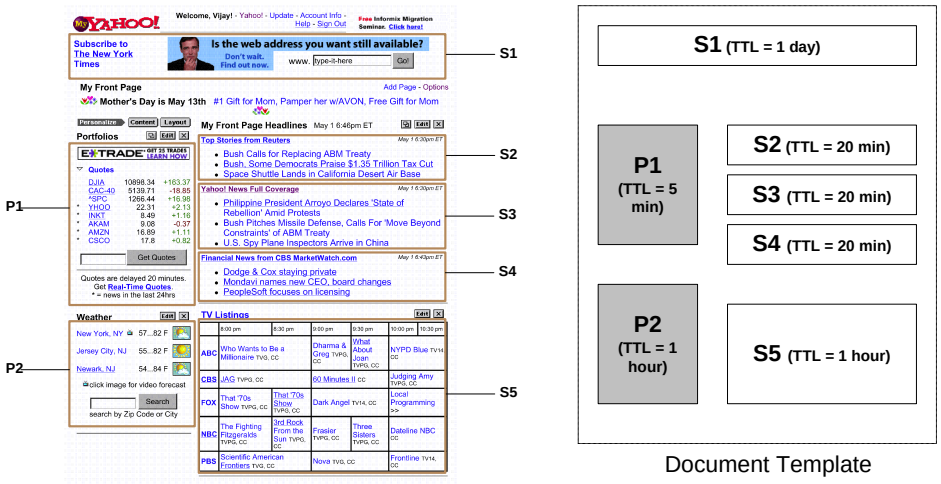


Figure 1: Dynamic content can be viewed in terms of a quasi-static document template and individual objects, which exhibit different sharing, cacheability, and freshness time characteristics.

fraction of the document that can be reused is in some sense a derived metric that depends on several independent parameters: the sizes of individual objects, their *freshness time* (the length of time the object remains valid), and the correlation between the two. The larger the content reusability, the more the likely benefits from techniques that either reuse or cache individual component objects.

Section 4 characterizes dynamic content from a representative set of web sites in terms of these independent and derived metrics.

3 Methodology

If one has access to the web server and/or application server of content providers, it is a relatively straightforward exercise to instrument these to obtain the above metrics of interest. Unfortunately, many commerical web sites which create dynamic content are proprietary. Therefore, we adopt an alternative approach based on the analysis of dynamic content (an HTML file) downloaded from various web sites.

The biggest difficulty that must be overcome in our chosen approach is the lack of an explicit template associated with the document, which can help identify its component objects. To work around this difficulty, we make the assumption that *the document template can be expressed as a nested table*. Figure 2(a) shows an example: here, the document includes head and body tags, with the body including 2 tables. We have been unable to find a web site whose pages are fully dynamically generated, where this assumption is not satisfied.

Given this assumption, our approach first extracts the template associated with the document, then identifies the (logical) objects that fill out this template (grouping neighboring objects that exhibit the same freshness time characteristics), and finally aggregates logical objects into physical objects that serve as the granularity at which document characteristics are modeled in Section 4. These steps are described in additional detail below:

1. **Data Collection:** In this first step, a dynamic document is downloaded every T time units from its web site to set up a series. Since the time interval used in the data collection determines the minimum value of the

freshness interval, this time interval should be set as small as possible. The documents in this series are denoted as $(d_1, d_2, \cdots, d_n)$.

2. **Tree Building:** Our assumption about the document structure allows the document to be represented as a tree of objects. In principle, this should be an easy affair: since HTML has an associated XML DTD associated with it, one could just build the object tree from the corresponding XML tree. However, we found that the HTML produced by many servers does not fully comply with the specification. Therefore, we first preprocess the document using `tidy` [28] to make it comply with the W3C XHTML 4.01 standard.

   Each leaf node in the tree is a possible object. To assist with the comparison step described next, each node in the tree is annotated with a *content digest*. The latter is computed using the collision-resistant SHA-1 hash function [1]. Internal nodes are associated with a hash of the content digests of their child nodes. Continuing with our example, the corresponding object tree of the document in Figure 2(a) is shown in Figure 2(b). Such object trees are computed for all document instances producing $(t_1, t_2, \cdots, t_n)$.

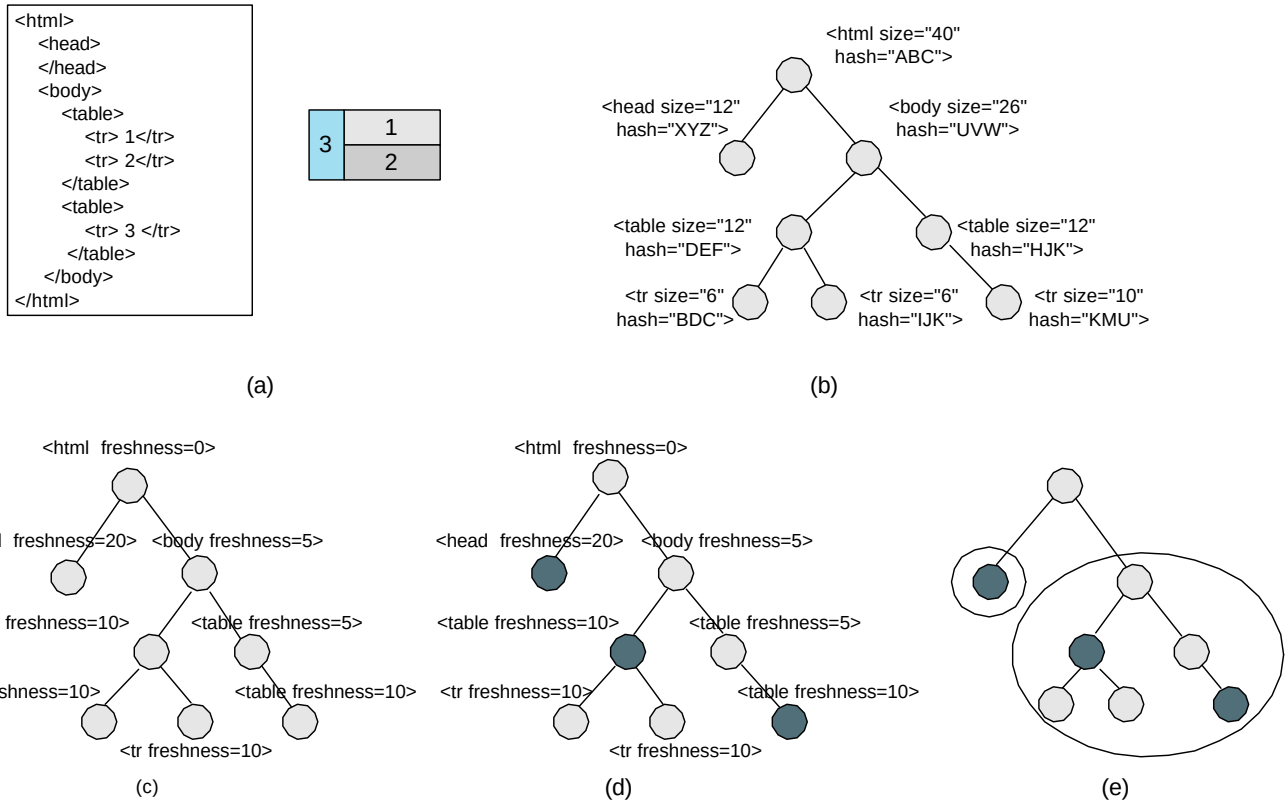


Figure 2: Identifying the objects making up a document involves five steps, going from (a) a simple document d, to (b) the corresonding object tree of document d, to (c) the object tree annotated with freshness times, to (d) logical groupings of objects, to (e) the final objects chosen to represent this document.

3. **Tree Comparison:** Based on the object tree constructed in Step 2, the freshness time of each object is computed by comparing the rest of the series with the first document t_1. The comparison is computed in a bottom-up fashion by comparing the digest of each node; the freshness time of the object is set to the time interval

corresponding to the earliest document where the digests differ. Figure 2(c) shows the results of this step, which is also represented as an XML tree.

4. **Object Grouping:** This step clusters contiguous objects (in the object tree) with similar properties into a single large logical object. The algorithm again proceeds in a bottom-up fashion from the leaf nodes: sibling nodes with the same freshness times are combined with each other, and a singleton (grouped) node is combined with its parent under the same conditions.

   Figure 2(d) shows the result of object grouping: the dark circles represent objects remaining as the leaf nodes. In this specific example, the first two `tr` nodes *1,2* are combined with their parent node `table`, while the third `tr` node exhibits a different freshness time than its parent node, and hence cannot be combined. These remaining leaf nodes define the component objects of the document, with the rest of the tree representing the template.

5. **Tree Splitting:** The last step of the object identification process groups one or more of the logical objects into physical objects; the latter represent the granularity at which surrogates and proxy caches work with the document objects. Ideally, this decision ought to be driven by the document semantics, taking into consideration factors such as freshness times, relationships between objects, etc.

   In the absence of such information, we have to use heuristic means. In this paper, we use two techniques to split the object tree generated in Step 4 into component objects: *size-based* and *level-based*. Size-based splitting chooses nodes whose size exceeds a certain threshold (and whose child nodes have size smaller than the threshold). Level-based splitting follows the logical structure of the document: all nodes below a certain depth are grouped together. Returning to our example, if only two levels are allowed, there will be only two nodes in the document, the `<head>` and `<body>`. Note that although we employ these two heuristics independently, they can be combined to choose document objects.

4 Analysis of Dynamic Content Characteristics

To model the characteristics of dynamic content, both in terms of independent parameters such as the number, sizes, and freshness times of objects, as well as derived parameters such as content reusability, we analyzed traces collected from six web sites using the methodology described in the previous section.

The main pages at these sites, which included three news sites (`www.cnn.com`, `dailynews.yahoo.com`, `www.nytimes.com`), two e-commerce sites (`www.amazon.com`, `www.barnesnoble.com`), and an entertainment site (`www.windowsmedia.com`), were downloaded every five minutes over a two-day interval.[1] Table 1 lists the dates during which the traces were collected. A point that needs to be made here is that we intentionally chose collection periods, where the web site was functioning in a "stable" mode. The primary reason for this choice is our perception that such information is more useful for longer-term surrogate and proxy cache planning as opposed to the unpredictable behavior of a web site during "unstable" periods (consider for example the nature of dynamic content that news sites generated in the period 09/10 - 09/12).

[1] We actually collected traces over a two-week duration for most of the sites, but these exhibit the same behavior as the two-day traces.

Site	cnn	yahoo	nytimes	amazon	bn	wmedia
Date	08/23 - 08/25	09/14 - 09/16	10/16 - 10/18	10/28-10/30	10/25-10/27	11/05-11/07

Table 1: Periods during which data traces were collected from various web sites. All periods are in the year 2001.

Each document was split into physical objects according to six different size limits (0, 200, 500, 750, 1000, and 2000 bytes), and three different level limits. We only report on a subset of these experiments in this section, primarily to improve the readability of the figures and tables. An additional point needs to be made here. In the size-based splitting scheme, it is possible to have some objects that are smaller in size than the specified limit. This comes about because we assume a static template structure across the document series, which has the consequence that whenever we mark a logical object as being a physical object we also need to apply the same marking for its siblings (even if their size is smaller than the limit).

4.1 Number of Objects

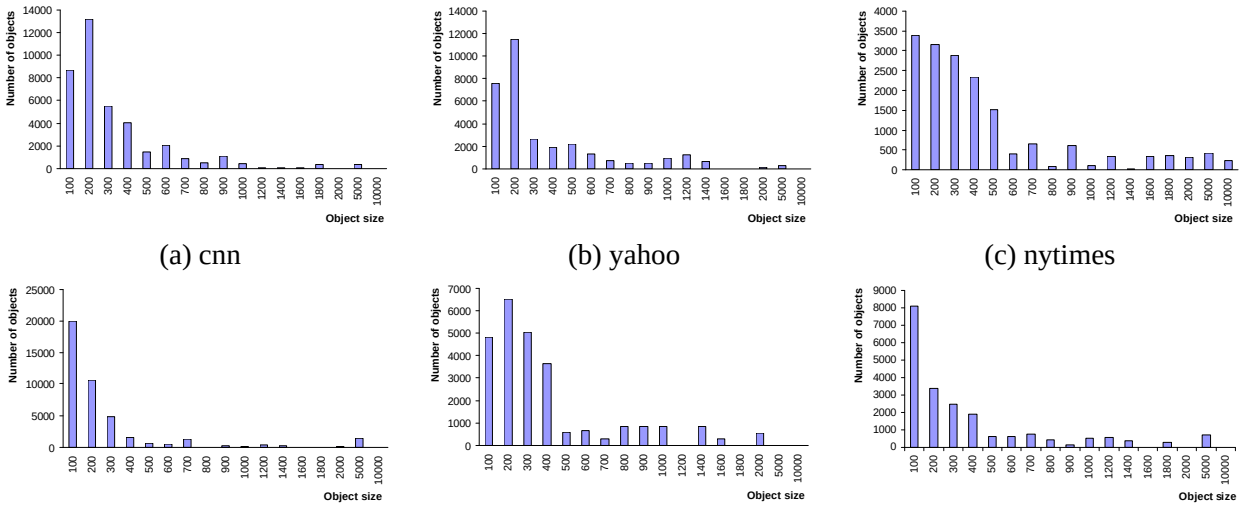
The number of objects in a dynamic document helps determine what opportunity, if any, is available for exploiting reuse. Unlike well-studied corresponding questions in the static content case, for example, the number of embedded images in a document [5], the number of physical objects in a document is influenced by means using which we choose objects. The average number of objects for each document is shown in Table 2. Each row in the table shows, for a fixed level limit, the number of chosen objects for different size limit settings. These counts verify intuition: the larger the size limit, the fewer the number of objects. Similarly, the deeper the chosen level, the more objects there will be.

Unless explicitly stated to the contrary, the measurements in the rest of the section refer to the largest level limit for each of the documents. We have experimented with lower limit settings and have found that the results overall have the same flavor. Note also that although the number of objects in a given document is sometimes small, considering these objects over the entire 2-day series provides a statistically significant sample.

4.2 Distribution of Object Sizes

Similar to the size distribution of static documents, we need to understand the size distribution of objects embedded in a dynamic web document. Many previous studies have verified that the size of a static resource, characterized by a heavy tail, is captured well by a pareto distribution (with $\alpha = 1.0 \sim 1.5$). However, intuition suggests that this characterization may not apply to objects within a dynamic document for two reasons: first, dynamic content itself tends not to have overly large sizes, and second, the size of an embedded object is restricted to be smaller than that of the overall document. Figure 3 shows the distribution of object sizes for our six document series, which verifies this intuition. We find that a large number (about 90%) of objects in our documents are of relatively small size (less than 500 bytes). This fact is better highlighted in the cumulative distribution function graphs shown in Figure 4.

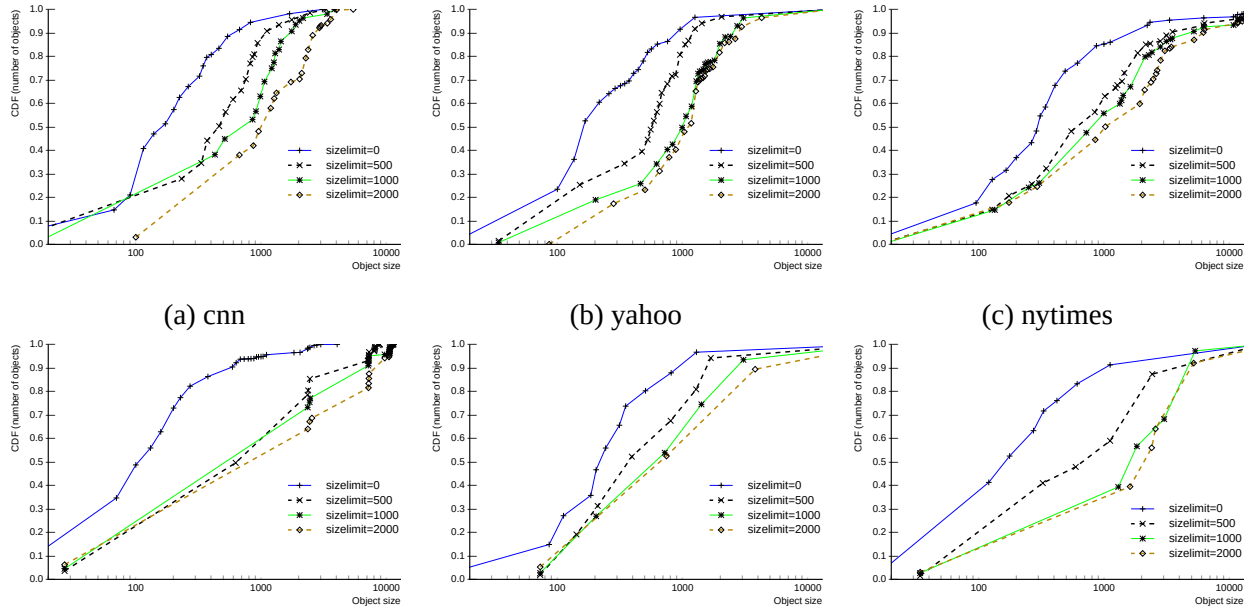
To develop a model for the object size distributions seen for different size limits, we use standard statistical methods similar to those used by Paxson et al. in [27]. We use both Chi-Square and Anderson-Darling (A^2) empirical distribution functions (EDF) for estimation of goodness-of-fit [12]. We found that most of object size distributions have a very good fit to the *Exponential* distribution, whose cumulative distribution function is $F(x) =$



(a) cnn (b) yahoo (c) nytimes

(d) amazon (e) bn (f) wmedia

Figure 3: The distribution of object sizes at different web sites assuming a size limit of 0. Each bar represents the number of objects contained in the size range bounded above by the value on the X-axis and below by the value of the bar to its immediate left.



(a) cnn (b) yahoo (c) nytimes

(d) amazon (e) bn (f) wmedia

Figure 4: The cumulative distribution function (of the number of objects) versus object size for different size limit settings.

Site	Level	Size Limit					
		0	200	500	750	1000	2000
cnn	level=18	142	90	71	59	49	33
	level=12	118	90	71	59	49	33
	level=6	35	35	32	30	30	26
yahoo	level=26	113	105	64	45	38	35
	level=20	113	105	64	45	38	35
	level=14	85	81	58	42	37	35
nytimes	level=23	69	53	34	30	29	23
	level=17	67	51	34	30	29	23
	level=11	63	49	32	28	28	23
amazon	level=21	178	55	28	26	22	16
	level=15	106	54	27	25	22	16
	level=9	54	31	22	20	20	16
bn	level=21	114	60	51	45	36	19
	level=15	80	45	38	34	30	19
	level=9	55	31	27	26	22	11
wmedia	level=19	73	40	30	26	17	15
	level=13	47	36	30	26	17	15
	level=7	17	14	14	13	13	12

Table 2: The mean number of objects corresponding to different size and level limits for the six web sites.

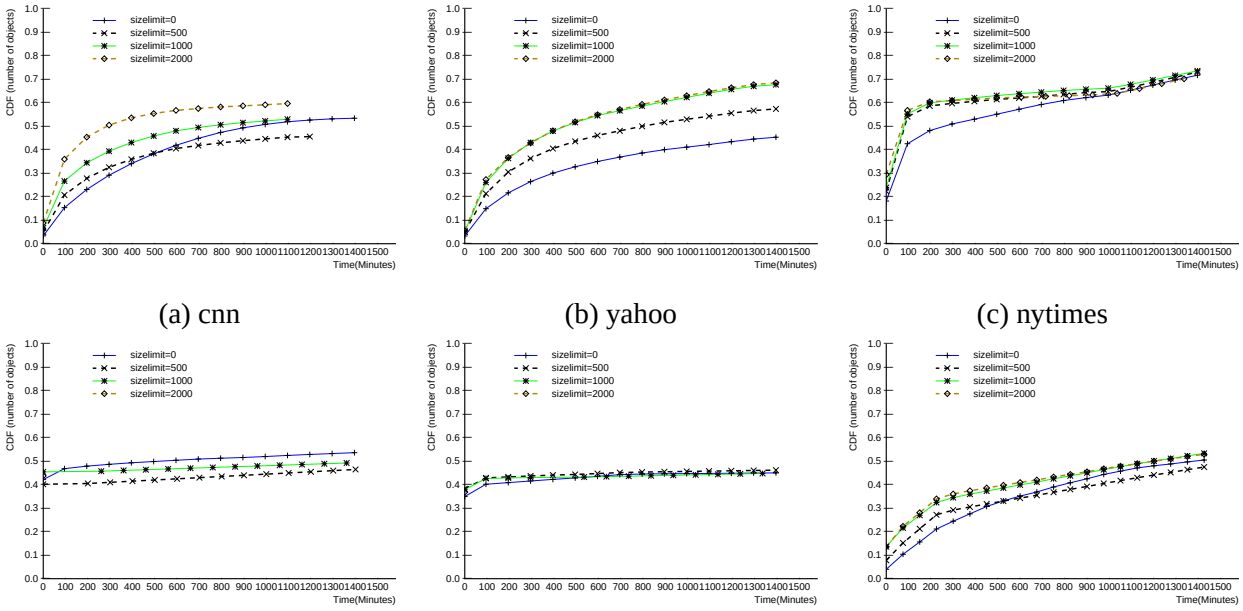
$1 - e^{-\lambda x}$. `amazon` and `wmedia` are two exceptions, exhibiting a slightly better fit to a *Weibull* distribution. We note that the Weibull distribution (with a CDF $F(x) = 1 - e^{-(\lambda x)^b}$) is a variant of the Exponential distribution, so these anomalies are likely not very crucial. Table 3 shows the parameters of the Exponential or Weibull distributions that best fit our trace data for various size limits. The results of the Chi-Square tests were in the range 0.1 to 2.5, showing a very close fit. The results of the A^2 test showed no significance in terms of goodness of fit for most of the documents; however, this failure, which is a common problem with EDF [5, 27], was primarily due to the fact that a fairly large data set was used. The fit improved significantly when we sub-sampled the data set. Although not shown, we observed similar distribution fits for different settings of the level limit.

Site	Distribution	Size Limit			
		0	500	1000	2000
cnn	Exponential	0.0040	0.0017	0.0011	7.3×10^{-4}
yahoo	Exponential	0.0034	0.0014	8.5×10^{-4}	7.8×10^{-4}
nytimes	Exponential	0.0024	8.5×10^{-4}	8.5×10^{-4}	8.5×10^{-4}
amazon	Weibull	$\lambda = 0.0058$ b = 0.8	$\lambda = 0.0011$ b=0.5	$\lambda = 7.68 \times 10^{04}$ b=0.6	$\lambda = 5.10 \times 10^{-4}$ b=0.6
bn	Exponential	0.0030	0.0015	0.0011	8.5×10^{-4}
wmedia	Weibull	$\lambda = 0.0037$ b = 0.8	$\lambda = 9.52 \times 10^{-4}$ b=0.7	$\lambda = 5.03 \times 10^{04}$ b=1.0	$\lambda = 5.03 \times 10^{-4}$ b=0.9

Table 3: Parameters of Exponential and Weibull distributions best fitting the measured object size distributions for the six web sites.

4.3 Distribution of Object Freshness Times

As described in Section 2, object freshness times are computed by comparing objects in a particular document against each subsequent document in the series. Each object is associated with the shortest time interval where the corresponding object in the template differs from the current one. Figure 5 shows the measured cumulative distribution function (of the number of objects) versus average object freshness times in the documents for different settings of size limit. A point (x, y) on the curve represents the number of objects (y) in the document that have a freshness time below x. The reason the CDF curve does not reach 1 at the extreme right of the plots is because the freshness time of the remainder of the objects is significantly higher than the right limit (of 1435 minutes $\sim$ 1 day). These latter objects essentially stay fixed throughout a document series.



(a) cnn (b) yahoo (c) nytimes

(d) amazon (e) bn (f) wmedia

Figure 5: The distribution of freshness time within different sizelimit.

The following observations can be made from the figures:

- As noted above, a large fraction of objects (in cnn's case, this number is about 40%) in all six of the documents do not change within the timescale of a day. In fact, the objects do not change even over timescales of a week, indicating that significant reuse can be exploited even by just focusing on objects that exhibit this relatively simple behavior.

- For those objects that do change within the timescale of a day, we found that a large fraction of objects have small freshness times, and that this fraction drops off as the freshness time increases. It would seem that such behavior can be captured by the *pareto* distribution ($F(x) = 1 - (k/x)^{-\alpha}, x > k$). However, similar to the observation made by other researchers [5], we found that although pareto is effective at capturing the heavy tail nature of the freshness time distribution, it does not fit well with the body, particularly for low freshness times. We then looked at the Weibull distribution, which has been previously used to model lifetimes of components

in computer systems, and found that it can characterize our measured behavior much better.

Table 4 shows the parameters of the Weibull distribution that best fit the data: with the exception of the commercial sites discussed below, our results show that the object freshness times can be effectively modeled using a Weibull distribution.

Site	Distribution	Size Limit			
		0	500	1000	2000
cnn	Weibull	$\lambda = 5.59 \times 10^{-4}$ b=0.6	$\lambda = 5.03 \times 10^{-4}$ b=0.5	$\lambda = 5.49 \times 10^{-4}$ b=0.4	$\lambda = 8.55 \times 10^{-4}$ b=0.3
yahoo	Weibull	$\lambda = 5.03 \times 10^{-4}$ b=0.7	$\lambda = 6.02 \times 10^{-4}$ b=0.5	$\lambda = 0.001$ b=0.5	$\lambda = 0.001$ b=0.5
nytimes	Weibull	$\lambda = 0.0011$ b=0.3	$\lambda = 0.0018$ b=0.2	$\lambda = 0.0021$ b=0.2	$\lambda = 0.0017$ b=0.1
amazon	Bimodal	$0 : 0.42$, $> 1435 : 0.46$	$0 : 0.40$, $> 1435 : 0.53$	$0 : 0.45$, $> 1435 : 0.5$	$0 : 0.43$, $> 1435 : 0.57$
bn	Bimodal	$0 : 0.35$, $> 1435 : 0.55$	$0 : 0.38$, $> 1435 : 0.54$	$0 : 0.38$, $> 1435 : 0.55$	$0 : 0.37$, $> 1435 : 0.58$
wmedia	Weibull	$\lambda = 5.03 \times 10^{-4}$ b=0.7	$\lambda = 5.03 \times 10^{-4}$ b=0.7	$\lambda = 5.03 \times 10^{-4}$ b=0.5	$\lambda = 5.03 \times 10^{-4}$ b=0.5

Table 4: Parameters of distributions best fitting the freshness time measurements for the six web sites. The two commercial sites show a degenerate case of the Weibull distribution: almost all of the objects have either near zero freshness time or very large freshness times. The fraction of objects falling into each category are denoted using the form $a : b$: a indicates the freshness time and b reflects the fraction.

- The freshness time distributions for the two commercial web sites (`amazon` and `bn`) markedly differ from that of the other sites in that all of the objects appear to fall into one of two categories corresponding to either a near-zero freshness time (accounts for about 40% of the objects) or those that have a very large freshness time exceeding one day. This behavior, which can be attributed to frequently changing dynamic advertisements included in the page, can be thought of as a degenerate case of the more general behavior that is captured using the Weibull distribution. Table 4 shows the fraction of the objects that fall into each category for the two sites corresponding to different size limits.

4.4 Correlation between Object Size and Freshness Time

To ascertain if there was a relationship between the sizes and freshness times of objects in the documents, we computed the correlation coefficient for the two series (see Table 5). While we had hoped to see a strong positive correlation between object size and freshness time (i.e., smaller objects exhibit smaller freshness times and vice-versa), we found that for our traces there was no conclusive indication of any correlation between the two metrics.

4.5 Extent of Content Reusability

As noted in the beginning of this paper, one of our main motivations for undertaking this study was to better understand the nature of content reusability in dynamic web content. Content reusability is a derived parameter, which

Site	Size Limit			
	0	500	1000	2000
cnn	-0.199	-0.280	-0.305	-0.477
yahoo	0.048	0.075	-0.013	-0.053
nytimes	-0.134	-0.196	-0.451	-0.549
amazon	-0.153	-0.900	-0.936	-0.999
bn	-0.064	-0.365	-0.329	-0.641
wmedia	-0.204	-0.254	-0.329	-0.316

Table 5: Correlation between object size and freshness time for the six sites, expressed in terms of the correlation coefficient.

is affected both by the distribution of object sizes and freshness times; however, this dependence is unpredictable, particularly in the absence of any correlation between the two.

There are two dimensions across which surrogates and proxy caches can benefit from content reuse. The first dimension, time, benefits repeated downloads of the same document: reuse in this case refers to the fraction of objects that remain valid across multiple downloads. The second dimension, document links, improves performance for downloading a group of linked documents. Reuse in this latter case refers to the fraction of objects that are shared across two (different) documents. We examine these two dimensions in turn.

4.5.1 Reusability across time

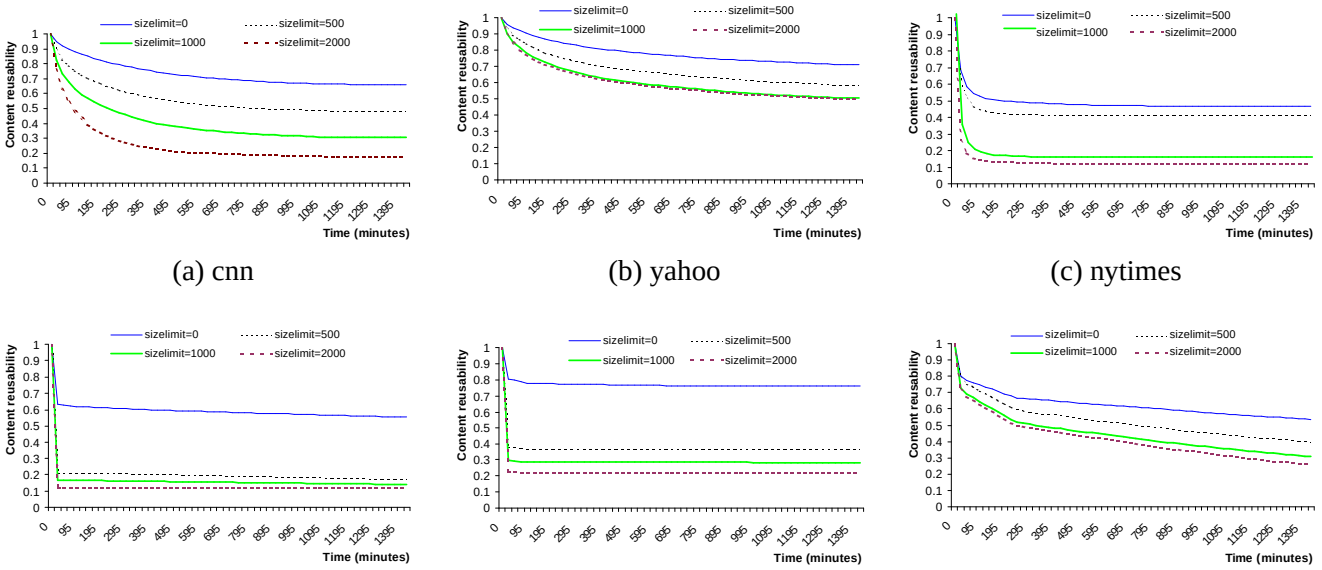
There are two ways to determine content reusability across time in the same document, based upon one's assumptions about the existence of a document template. The first, referred to hereafter as *content reusability*, assumes that each document is associated with a static template, and defines an object to be reusable *only* if it occupies the same place within the template. The second, which we call *object reusability*, takes a more general view in defining an object to be reusable if it appears *anywhere* in a new document.

Figure 6 shows the content reusability (in terms of the bytes in the document that can be reused) over a one-day interval for the homepages of our six web sites. A few points need to be made here. First, each curve is an average of 288 contiguous documents (corresponding to a one day trace spaced at five minute intervals), each of which has been compared with 288 subsequent documents in the series. Second, for the measurements reported here, the level limit was set to the highest possible in each case to ensure that we obtained the maximum potential content reusability. Finally, the curves include reuse of the template, which occupies between 10%–20% of the whole document.

Several observations can be made from these figures:

1. The smaller the granularity of objects, the more the reusability, which complies with our intuition. For example, 75% of the document bytes are resuable in the case of cnn after 12 hours have elapsed when the size limit for defining objects is set to 0 (i.e., there are no restrictions on how small an object can be), while the corresponding reusability is 20% when the size limit is set to 2000 bytes.

2. Content reusability stabilizes after a time period, and thereafter stays flat for a long time. We redid these measurements using two-week traces but the results stayed the same, indicating that a certain fraction of the document remains usable over extended periods of time.



(a) cnn (b) yahoo (c) nytimes

(d) amazon (e) bn (f) wmedia

Figure 6: Content reusability of different web sites averaged over a one day period, assuming a static document template.

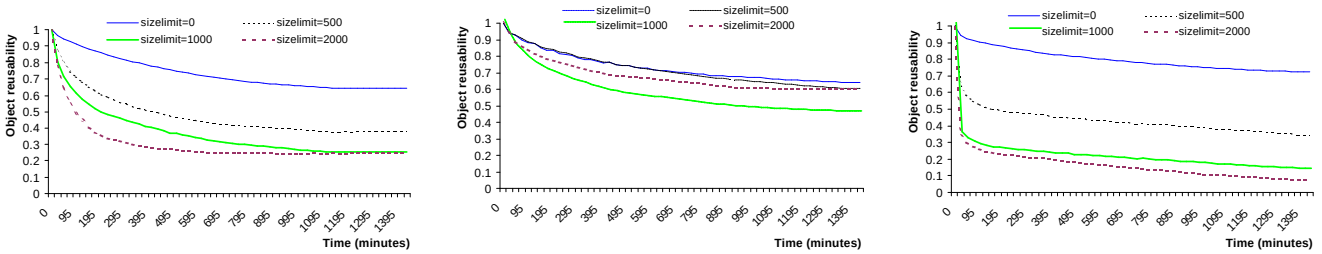
3. The content reusability curves for the two commercial web sites are flat, corresponding to the bimodal nature of their freshness time distributions. It is only possible to reuse the unchanged portions of the document, since the rest changes too frequently for there to be any reuse.

4. For the remaining four sites, acceptable levels of reusability (larger than 40%) require that the size limit of objects be set less or equal to 500 bytes. However, what is encouraging (from the perspective of someone trying to design a proxy cache system) is that the amount of content reusability degrades gracefully as the size limit is increased. Thus, there is a range of size limits, based on the efficiency and overheads associated with caching at the object granularity, where a proxy cache can see benefits from reuse.

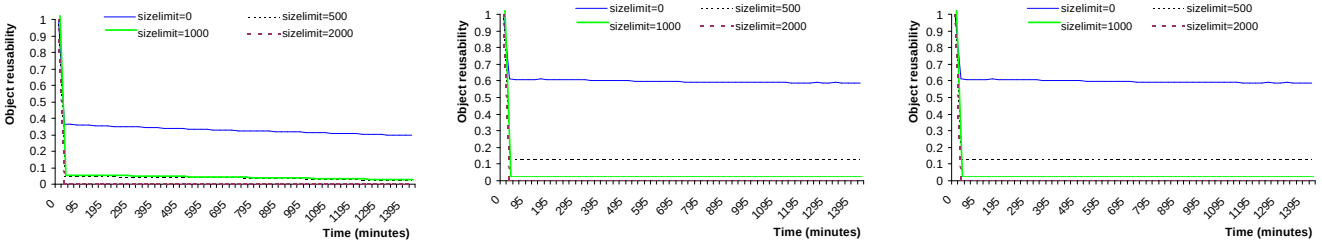
Figure 7 shows the corresponding figures for object reusability. As noted earlier, this represents the fraction of the document bytes that can be reused after a certain time period, assuming that an object in the first document appears somewhere in the second. Intuitively, object reusability should be greater than content reusability. However, while comparing the set of objects, we do not consider the document template and objects that are smaller than a certain threshold. Table 6 shows the fraction of document bytes accounted for by the objects that are considered. As we can see, the unaccounted for objects end up contributing a sizeable fraction of the overall document bytes and because they are deemed not reusable, object reusability ends up being slightly lower than the corresponding content reusability. Other than this difference, object reusability exhibits the same characteristics as content reusability and the same observations apply.

4.5.2 Reusability across linked documents

Content reuse is also possible across dynamically generated documents from the same site that end up, for various reasons, to share the same set of objects. If these documents are linked to each other, then one can exploit this reuse



(a) cnn (b) yahoo (c) nytimes



(d) amazon (e) bn (f) wmedia

Figure 7: Object reusability of different web sites averaged over a one day period, assuming the document does not have a template.

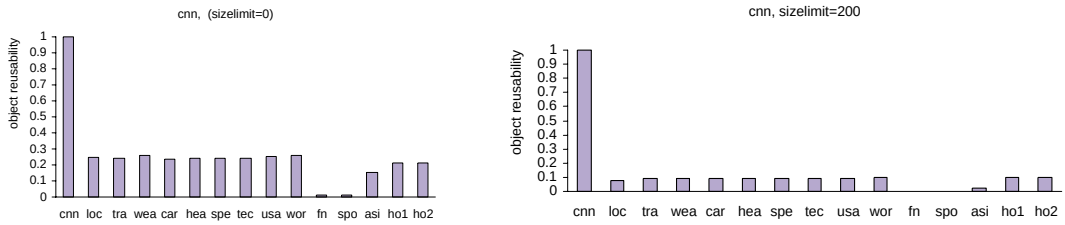
Site	Size Limit					
	0	200	500	750	1000	2000
cnn	89.4%	85.0%	73.5%	76.4%	60.3%	64.2%
yahoo	93.5%	83.0%	87.6%	83.7%	73.0%	50.8%
nytimes	96.4%	93.5%	88.0%	90.8%	88.0%	79.3%
amazon	82.4%	87.3%	93.3%	90.7%	90.9%	93.4%
bn	98.2%	98.0%	96.4%	96.4%	96.2%	96.2%
wmedia	91.0%	90.8%	91.8%	91.3%	88.8%	89.7%

Table 6: The fraction of the document bytes accounted for by objects selected in the object reusability computation.

across traversals of the links. As an example, consider the `cnn` web page, which contains links to pages that contain additional detail about headlines, as well as news in various categories. Given that all of these pages are generated on the same server, it appears reasonable that they would share some objects.

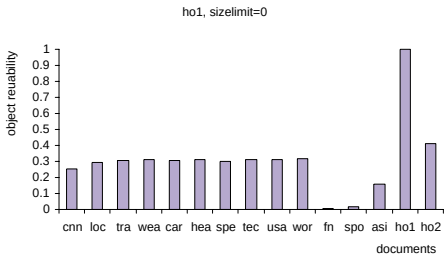
To verify this intuition, as part of our trace collection process, we also collected a set of documents that were linked in from the home page of each of our sites. To take `cnn` as an example, we collected fifteen other documents in addition to the home page: `local`, `travel`, `weather`, `career`, `health`, `special`, `technology`, `US`, `world`, and the top two headlines, all of which are hosted on the CNN server, as well as two links `sports` and `asia`, which are hosted on different servers (`sportscnn.cnn.com` and `asia.cnn.com`).

Although these 15 documents do not share the same document template, our analysis results show that large parts of them are in fact shared. Figure 8 shows the object reusability across the different documents, using as reference the document from a particular page. For instance, Figure 8 (a) and (b) show the reuse with respect to the main `cnn` page when the size limit is 0 bytes and 200 bytes respectively. For the smaller size limit, we see that 25% of the document bytes can be shared across other documents that are also hosted on the same server, while the sharing

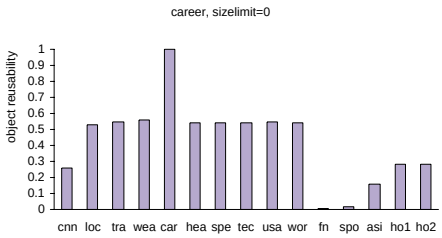


(a) cnn vs. others

(b) cnn vs. others (limit=200)



(c) hot news vs others



(d) career vs. others

Figure 8: Object reusability across documents from the `www.cnn.com` web site with respect to (a) the `cnn` default page (with size limit 0); (b) the `cnn` default page (with size limit 200); (c) a headline page; and (d) the career news page.

across documents that are hosted elsewhere is not so good. This verifies our intuition about server activity. The measurements for the larger size limit shows that achieving these gains requires the ability to reuse objects at a fine granularity.

Figure 8 (c) and (d) show the corresponding measurements using one of the headline pages and the career page as a reference. We find that the amount of reusability is higher (between 40%–50%), primarily attributable to the smaller size of the overall document in these cases. Thus, our results verify that there is substantial opportunity for reusing objects across dynamic documents that are linked from each other.

4.6 Implications

Our study of object characteristics of dynamic web content has several implications for future research in optimizing the generation and delivery of dynamic web content:

- An immediate implication, which we explore in the rest of this paper, is that such models enable the development of synthetic content generators, a prerequisite for simulation-based studies of the kind that have previously proved very successful in the static content case.

- Our results also answered most of the questions we posed in Section 1. Specifically, our results show that there is a significant opportunity for content reuse, although one needs to distinguish between content that exhibits a graceful degradation of reusability with object size (e.g., `cnn`, `yahoo`, `nytimes`, and `wmedia`) and the kind that exhibits a sharper degradation (e.g., `amazon` and `bn`). That some dynamic content exhibits the first kind

of behavior is encouraging: it frees up a surrogate or proxy cache designer to work with a range of object sizes in order to achieve benefits from reusability.

- The Weibull nature of the freshness times distribution suggests that because a large number of objects get updated very frequently, client-initiated cache consistency protocols as opposed to server-initiated ones such as volume leases [34], may be more appropriate for these dynamic objects.

- Knowledge of object size and freshness time distributions can be exploited by a cache designer who can incorporate them into heuristics to decide which objects to cache and which to refetch. For example, based on the caching history, an object can be tagged as being more likely to provide an increased reuse opportunity if it has already been reused a certain number of times.

5 DCE: Dynamic Content Emulator

Although several researchers have proposed server-side and cache-side techniques for improving generation and delivery of dynamic content, these techniques have not as yet been widely adopted by either content providers or content deliverers. We attribute this situation to the following: content providers need to be first convinced of the benefits of such techniques before they modify their content to support composition and reuse at the granularity of sub-document objects. Unfortunately, it is difficult to verify the efficacy of a proposed technique in the absence of either real or synthetic content that adheres to the consensus view of dynamic content as consisting of a document template and individual objects.

In this section, we describe the design of a dynamic content emulator (DCE), which alleviates the above situation. DCE uses the models of object size and freshness time obtained in the previous section to emulate a web server that serves dynamic content, which incorporates support for document templates and identification of component objects.

5.1 Design

DCE separates out the functions of content generation and content representation into two modules: the *dynamic content generator* (DCG) and the *dynamic content presentor* (DCP). DCP interfaces with requesters and appears as a traditional web server augmented with additional functionality described later in the section (see Figure 9).

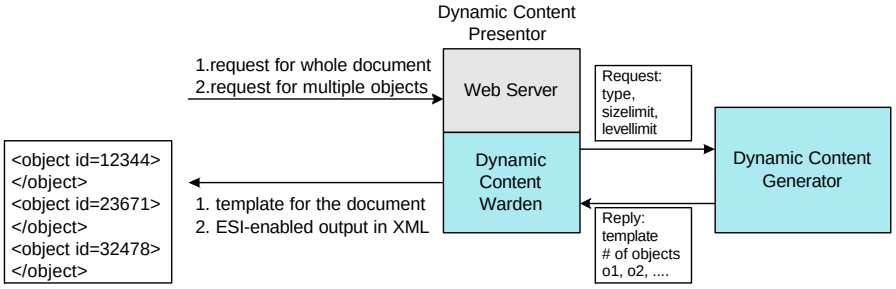


Figure 9: The architecture of dynamic content emulator (DCE), which includes two components: a dynamic content generator (DCG) and a web server-like dynamic content presenter (DCP). The extra functionality of DCP is provided by a module called the dynamic content warden.

To simplify its use, DCP supports a fairly simple interface, servicing two kinds of requests, either for the *document template* or for a set of *selected objects*. In response to requests of the first kind, the DCP checks a local cache to see if the document template exists, and if it is still valid. If both conditions are satisfied, it replies with the template as described below. If not, it requests the DCG to generate this document. The DCG takes three parameters as input to create the corresponding document template and individual objects: the *dynamic content type*, the *size limit* and the *level limit*. The latter two parameters have the same meanings as discussed earlier, while the first one is used by the DCG to index into a database of canned distributions. For example, the request $< cnn, 200, -1 >$ results in the DCG generating a `cnn`-like document (i.e., with the characteristics reflected in the `cnn` trace) with objects larger than 200 bytes and a level limit set to infinite.

Requests for a single object are handled similarly. The DCP first checks the cache to verify validity, and if the object freshness time has expired, then it requests the DCG to get a new version. The DCG services this request by looking up the distribution corresponding to the document template. The DCP extends this basic scheme to support requests for a set of objects.

5.2 Implementation

Our implementation builds the DCP on top of the Jakarta Tomcat 4.0 web server infrastructure. All functionality for dealing with dynamic content (understanding templates, verifying validity times, etc.) is encapsulated in a *dynamic content warden*, which is implemented as a servlet. The DCG is implemented as a standalone Java application, and interfaces with a database that contains parameters for modeling various kinds of dynamic content. Configuration parameters associated with the dynamic content warden allow simulation of multiple web sites, and selection of different object granularities for a particular site. Both the DCG and the dynamic content warden are easy to extend to incorporate different representations of the document template and different distributions governing object size and freshness time characteristics.

The content presented to requesters by the DCP is currently packaged as an HTML document augmented with *Edge Side Include (ESI)* directives. We considered using the *XSL-FO* formatting language from W3C, however felt that because of its complicated nature it would be a while before content providers and caching vendors would support this standard. ESI, on the other hand, benefits from the fact that it is a natural extension to Server-Side Include (SSI), which is widely adopted.

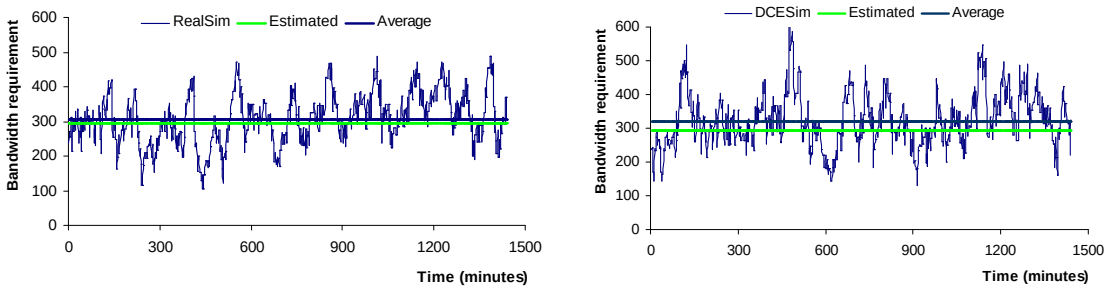
Our implementation had to extend the ESI specification in two ways. These extensions, which concern requests and responses that involve multiple objects within the same document, might make sense to be included in the ESI specification as it evolves. Logically, to support ESI, downstream surrogates and/or proxy caches will send requests for all of the objects that are invalid in the local cache. Although these requests can be sent sequentially, the absence of widespread support for HTTP 1.1 implies that one is likely to see sizeable overhead for creating a different connection for each request. An alternative, which we have adopted, combines the requests into a single message. Our suggestion for the combined requests is the comma separated values (CSV) format. For example, the request for three objects in our example might look like `http://128.122.142.137:8080/dcg/pageServer?object=12344,23671,32478` where the last three numbers refer to unique IDs of objects. We also suggest using XML to describe the reply for mutiple objects (see Figure 9). The second extension associates freshness times with individual objects (instead of the entire document) by adding an attribute to the <`esi:include`> tag, which now

looks at <esi:include freshness='8' src='URL'>.

5.3 Validation of DCE

To validate both the DCE, and as a consequence our analysis models, we developed an idealized proxy cache simulator that can be driven either from our collected traces or from emulated content generated by the DCE. The output of the simulator is the number of response bytes transmitted between the server and the cache, assuming that requests for the web page of interest arrive at the cache according to a poisson arrival model with mean $\lambda = 1$ minute.

When driven by real trace data, the simulator computes the transferred bytes by comparing the currently cached document against the document corresponding to the request time, inferring the objects that have changed (using a method similar to that described in Section 2), and replaces the cache contents with the new document. When driven by DCE, the simulator maintains a cache of the document objects and requests new versions upon detecting expiration: these requests are handled by the DCG, which looks up the database of distribution parameters (corresponding to the trace data) to send an object with size and freshness properties randomly drawn from the corresponding distributions. Figure 10(a) and 10(b) show the results of the two cases while simulating client activity over a one day period: RealSim refers to the simulation using the trace data, while DCESim refers to the simulation using DCE.



(a) (b)

Figure 10: The bandwidth requirements in RealSim and DCESim. The plots also show the predicted value of bandwidth, obtained as discussed in the text.

As Figure 10 shows, the DCE-driven simulation is able to capture the behavior of the real trace data: the mean bandwidth requirements in the two cases are identical, as are the lower and upper bounds of required bandwidth. More importantly, and this is something that the figure does not show, DCE offers a significant advantage in simulation time attesting to its utility as an efficient tool for evaluating new techniques. In particular, to simulate the one day interval on a dual Pentium III-1GHz machine with 512 MB of memory, the simulator took 0.5 seconds when driven by DCE, in contrast to the 2.5 minutes that it took to process the real trace.

Figure 10 also highlights an additional advantage of modeling the characteristics of dynamic content. In some situations, as here, these models can be used to obtain a quick analytical estimate of the values of interest. For example, we can use the following simple prediction model to estimate the average bandwidth requirement between the proxy cache and the server. Assuming that the document consists of N objects (o_1, o_2, $\cdots$,o_N), with sizes and freshness times $(s_1, s_2, \cdots, s_N)$ and $(f_1, f_2, \cdots, f_N)$ respectively, the average bandwidth requirement seen by an idealized proxy cache which receives requests according to a poisson-arrival model with mean λ is given by the

following expression:

$$BW = \sum_{i=1}^{M} s_{z_i} \approx N \cdot \bar{S} \cdot F(\lambda) \quad (1)$$

where M is the number of objects that have expired when the request comes in, and $(z_1, z_2, \cdots, z_M)$ are the sequence numbers of those objects. F is the cumulative distribution function of the freshness times $(f_1, f_2, \cdots, f_N)$ and $\bar{S}$ is the mean of the object sizes $(s_1, s_2, \cdots, s_N)$. The approximation comes about because an average request, that arrives every λ time units, finds $N \cdot F(\lambda)$ objects whose freshness times have expired ($F(t)$ represents the fraction of objects with freshness time less than t). One can assume that each of these objects has an average size of $\bar(S)$, resulting in the expression above.

For the trace data simulated in this experiment, which consisted of $N = 137$ objects with an object size distribution given by Exponential ($\lambda = 0.004$) and a freshness time distribution given by Weibull(F(λ) = 0.0086), the estimated bandwith works out to be 294. As shown in Figure 10, this estimate closely approximates the bandwidth requirements seen by both RealSim and DCESim.

6 Related Work

Web workload characterization has been extensively studied in the past five years from the perspective of proxies[6, 33, 13, 31], client browsers[11, 4, 20], and servers[3, 25]. However, many of these studies focus on the characteristics of web resources at the granularity of the whole document, such as content type, resource size, response size, resource popularity, modification frequency, temporal locality, clent access pattern, and the number of embedded resources. Many of these previous research results accurately capture the characteristics of static web content. However, for dynamic web content which introduces the notion of object composition, many of these characteristics, such as request and response sizes need to be revisited. Moreover, dynamic content necessitates understanding of new characteristics, such as the number and sizes of objects making up a document, the freshness times of these objects. To the best of our knowledge, the work described in this paper is one of the first efforts trying to model these latter characteristics.

Wills and Mikhailov [32] quantitatively analyzed the content reusability present in traces collected from several web sites after a one day interval. Two notable difference of our characterization include the characterization of content reusability at finer granularity (making explicit the time dependence), and the relationship between object size and content reusability. Our results indicate the object granularity that must be supported in order to successfully take advantage of potential reusability.

Challenger et al. [9] analyzed object size distributions based on server traces from the 2000 Olympics site. Although related to our objectives, their work works with a different definition of what an object is: they include not only the individual objects embedded within a document, but also the entire generated document itself. Because of this reason, while a pareto distribution fits their findings, we conclude that the sizes of individual objects in the document follow an exponential distribution.

Our work on the dynamic content emulator (DCE) resembles work done by Barford et al. on the SURGE static workload generator [5]. However, unlike SURGE, which was designed to model client access patterns, DCE focuses on the complementary goal of emulating server behavior, both in terms of its load properties as well as the nature of

the content itself. To simulate an appropriate delay to model overheads of dynamic content generation, our emulator uses delay models from previous research [19, 26, 17, 2, 21].

The work in this paper was motivated in part by our inability to extend, to our specific setting, the results previously obtained by researchers working on various aspects of dynamic and personalized content delivery. Such work, which has focused on both server-side [8, 9, 35] and cache-side [7, 14, 23, 32, 24] has typically been validated with specific, proprietary workloads. For example, Challenger et.al used the 1998 Olympic winter games workload in [8], and the 2000 Olympic games workload in [9], and Douglis et.al used a modified internal AT&T web-based "recruiting database" to evaluate their idea of HPP [14]. The work in this paper addresses this shortcoming, providing both models and tools that allow researchers to efficiently evaluate their techniques on a variety of (synthetic) workloads. We must note that our intention is not to replace experimentation with real workloads, but instead to complement it.

The work reported in this paper represents a particular view of dynamic content, which currently dominates traffic from real web sites. With proposed techniques such as the class-based page classification scheme in [35], the Active Cache system proposed by Cao et al. [7], and more recently the Gemini system proposed by Myers et al. [24], which require web servers to provide both content and specialized code (applets in Active Cache, Java classes in Gemini) to proxy caches, it is clear that the nature of dynamic content and therefore its characteristics will evolve in the future. While this may require that we revisit some of our results, we believe that the methodology we have adopted is sound and can be used to characterize these new forms of content as and when they become available.

7 Conclusions and Future Work

This paper has proposed a methodology for evaluating characteristics of dynamic web content, and used this methodology to obtain models for various independent and derived metrics of interest such as object sizes, freshness times, and content reusabililty. To summarize our main findings, we have found that (1) object sizes in dynamic documents can be modeled using an Exponential distribution; (2) for document objects that change over time, their freshness times can be modeled in terms of a Weibull distribution, which sometimes degenerates to a case where all of the objects change on every access; (3) there is significant opportunity for object reuse across both multiple accesses of the same document as well as accesses of related documents. Our results have also made explicit the dependence between the size at which document objects are managed and the corresponding reuse potential.

These models have also served as the foundation for the design of a tool, the Dynamic Content Emulator (DCE), which emulates a web server serving dynamic content, both in terms of the load properties as well as the nature of the generated content. DCE incorporates emerging standards such as ESI, permitting its use for evaluating object composition-based optimizations for improving dynamic content delivery at the level of surrogates or caches.

Our future work consists of using DCE to evaluate and further refine the design of our CONCA prototype [29], which incorporates a novel design for efficient caching of dynamic and personalized content.

References

[1] FIPS 180-1. *Secure Hash Standard*. U.S. Department of Commerce/N.I.S.T., National Technical INformation Service, Springfield, VA,, apr 1995.

[2] V. A. Almeida and M. A. Mendes. Analyzing the impact of dynamic pages on the performance of web servers. In *Proceedings of the Computer Measurement Group Conference*, December 1998.

[3] M. Arlitt and C. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 1997.

[4] P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella. Changes in web client access patterns: Characteristics and caching implications. *World Wide Web, Special Issue on Characterization and Performance Evaluation*, 2:15–28, 1999.

[5] P. Barford and M. E. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of Performance '98/ACM SIGMETRICS '98*, July 1998.

[6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. of the Conference on Computer Communications (Infocom'99)*, March 1999.

[7] P. Cao, J. Zhang, and K. Beach. Active cache: Caching dynamic contents on the web. In *Proc. of IFIP Int'l Conf. Dist. Sys. Platforms and Open Dist. Processing*, 1998.

[8] J. Challenger, A. Iyengar, and P. Dantzig. A scalable system for consistently caching dynamic web data. In *Proceedings of Infocom'99*, March 1999.

[9] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed. A publishing system for efficiently creating dynamic web content. In *Proceedings of INFOCOM'00*, March 2000.

[10] IBM Corp. Websphere platform. In *http://www.ibm.com/websphere*.

[11] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[12] R. B. D'Agostino and M. A. Stephens, editors. *Goodness-of-Fit Techniques*. Marcel Dekker, Inc, 1986.

[13] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of changes and other metrics: a live study of the world wide web. In *Proceedings of USITS'97*, 1997.

[14] F. Douglis, A. Hario, and M. Rabinovich. HPP:HTML macro-pre-processing to support dynamic document caching. In *Proceedings of USITS'97*, 1997.

[15] R. Doyle, J. Chase, S. Gadde, and A. Vahdat. The trickle-down effect: Web caching and server request distribution. In *Proc. of the 6th International Workshop on Web Caching and Content Distribution (WCW'01)*, June 2001.

[16] W3C XSL Working Group. http://www.w3.org/style/xsl/.

[17] Y. Hu, A. Nanda, and Q. Yang. Measurement, analysis and performance improvement of apache web server. In *Proceedings of the IEEE International Performance, Computing, and Communications Conference*, February 1999.

[18] Akamai Technologies Inc. Edgesuite services. In *http://www.akamai.com/html/en/sv/edgesuite_over.html*.

[19] A. Iyengar, J. Challenger, D. Dias, and P. Dantzig. High-performance web site design techniques. *IEEE Internet Computing*, 4(2), March/April 2000.

[20] T. Kelly. Thin-client web access patterns:measurements for a cache busting proxy. In *Proc. of the 6th International Workshop on Web Caching and Content Distribution (WCW'01)*, June 2001.

[21] B. Kothari and M. Claypool. Performance analysis of dynamic web page generation technologies. In *Proceedings of the International Network Conference (INC)*, July 2000.

[22] B. Krishnamurthy and J. Rexford. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching and Traffic Measurement*. Addison-Wesley, Inc, 2001.

[23] M. Mikhailov and C. E. Wills. Change and relationship-driven content caching, distribution and assembly. Technical Report WPI-CS-TR-01-03, Computer Science Department, WPI, March 2001.

[24] A. Myers, J. Chuang, U. Hengartner, Y. Xie, W. Zhang, and H. Zhang. A secure and publisher-centric web caching infrastructure. In *Proc. of the IEEE INFOCOM'01*, April 2001.

[25] V. N. Padmanabhan and L. Qiu. The content and access dynamics of a busy web site: Findings and implications. In *ACM SIGCOMM'2000*, 2000.

[26] V. Pai, P. Druschel, and W. Zwaenepoel. Flash: An efficient and portable web server. In *Proceedings of the 1999 USENIX Annual Technical Conference*, June 1999.

[27] V. Paxson. Empirically-derived analytic models of wide area TCP connections. *IEEE/ACM Transactions on Networking*, 1994.

[28] D. Raggett. HTML tidy tool. In *http://www.w3c.org/People/Raggett/tidy*, 2000.

[29] W. Shi and V. Karamcheti. CONCA: An architecture for consistent nomadic content access. In *Workshop on Cache, Coherence, and Consistency(WC3'01)*, June 2001.

[30] M. Tsimelzon, B. Weihl, and L. Jacobs. Esi language sepcification 1.0. In *http://www.esi.org*, 2000.

[31] D. Wessels. *Web Caching*. O'Reilly, 2001.

[32] C. E. Wills and M. Mikhailov. Studying the impact of more complete server information on web caching. In *Proc. of the 5th International Workshop on Web Caching and Content Distribution (WCW'00)*, 2000.

[33] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. In *Proc. of 17th ACM Symposium on Operating Systems Principles (SOSP)*, 1999.

[34] J. Yin, L. Alvisi, M. Dahlin, and C. Lin. Hierarchical cache consistency in a WAN. In *Proceedings of the second USITS*, October 1999.

[35] H. Zhu and T. Yang. Class-based cache management for dynamic web content. In *Proceedings of INFOCOM'01*, April 2001.