Geographic Web Usage Estimation By Monitoring DNS Caches

Hüseyin Akcan
CIS Department
Polytechnic University
Brooklyn, NY 11201
hakcan01@cis.poly.edu

Torsten Suel
CIS Department
Polytechnic University
Brooklyn, NY 11201
suel@poly.edu

Hervé Brönnimann
CIS Department
Polytechnic University
Brooklyn, NY 11201
hbr@poly.edu

ABSTRACT

DNS is one of the most actively used distributed databases on earth, accessed by millions of people every day to transparently convert host names into IP addresses and vice versa. In order to improve their performance, DNS servers also keep temporary records of all requested domain names in their cache. While most of the DNS servers are configured to be used by their local users only, there still exist many DNS servers that respond to public queries. Querying these DNS servers reveals the recently visited domains. Exploiting the geographically distributed nature of DNS, one can gather usage statistics ranging from a single DNS server to global scale. In particular, this enables collecting statistics about geographic differences in web browsing behavior between different regions of a country or the world. In this paper, we present methods to identify these public DNS servers, discuss how to effectively crawl them, and describe our algorithm to extract usage estimations from the crawl data. We also evaluate our estimation algorithm using extensive simulations, and finally use our algorithms to crawl 150 U.S. universities for various domains, and explore the effects of location and time on the access rate of these domains.

Categories and Subject Descriptors: H.2.8 Database Management: Database Applications - Data mining

General Terms: Measurement

Keywords: DNS, web site usage estimation, web access monitoring

1. INTRODUCTION

The web is one of the biggest sources of freely available information in the world. Although free, the information is mostly scattered, and requires specialized algorithms and often expensive infrastructure to be useful to us. Even though the majority of the work today focus on harnessing web content, there is also the other side of the coin, which is how this data is accessed by the users. The web usage statistics contain valuable information about the location and the demographics of the users. Except the aggregated data from a handful of companies, web usage statistics are only available locally to the website administrators. Alexa [1] was one of the first companies to provide statistics gathered from users via a toolbar. The toolbar is installed as a plugin to the existing web browsers of participating users, and reports every visit of the user to Alexa. The toolbar approach has important drawbacks; e.g., privacy issues limit the scalability of the approach in general, while on the contrary the popularity of the toolbar in some countries or businesses affects the randomness of the sample and introduces bias towards certain topics or geographical regions. One good example for this is the well known bias of Alexa results towards sites visited by people in the Search Engine Optimization (SEO) industry because SEO people are more likely to use the Alexa toolbar[1]. Getting the access statistics directly from the ISP logs is another approach for estimation, used by Compete [2], Hitwise [4], and Quantcast [5]. Even though the estimation in this case is more accurate, special business agreements are required to get private user data from ISPs, which makes it hard to scale (e.g., outside of U.S.). The limited size of the sample affects the accuracy of the estimates for both of these approaches, and only highly popular websites are accurately represented.

Fortunately in addition to the methods mentioned above, the web in itself contains traces of the user access data. In this paper, we focus on the problem of estimating the access rate of a website over time, but we use methods that are quite different than the traditionally used ones. The crux of our approach is to monitor the DNS caches in order to get a statistically reliable estimation about the access rate of individual domains. The estimations we get range from micro scale (single DNS cache) to global scale (country or larger), and cover a larger user base and a larger geographic area quite naturally, compared to the existing methods. For example, in a recent crawl in the Swiss IP domain, we identified $30,000$ responsive DNS caches.

Before going into the technical details, we would like to elaborate on the possible applications of our method. The main benefit of our approach is to give researchers a scalable way to gather independent data about the way users access the web without using any expensive infrastructure. Our

LocWeb 2008, April 22, 2008, Beijing, China.
LocWeb2008 held at WWW2008, April 22, 2008, Beijing, China.
ACM 978-1-60558-160-6/08/04 . . . $5.00.

[1]http://norvig.com/logs-alexa.html (logs from year 2006)

method is abstract enough to support a wide range of applications. One such application is to observe the browsing behavior of users in a certain location. Thus, by monitoring the relative popularity of particular web sites in a particular area, we could make conclusions about the demographics of the user population in that area, for example by estimating the frequency of visits say to popular Chinese or Turkish web sites. The gathered data is useful for research, as well as other applications such as advertising. The ability of our method to scale from a single DNS server to wider geographical regions makes the applications even more interesting. Apart from the location, one other dimension of the observations is the time. By extending the duration of the observations, we can get information about how the local browsing habits of users change with location, as well as time. Example applications for this approach include monitoring the access rate of the candidates' websites before election times to aid future predictions about the outcome of the elections, or to monitor competing websites to observe how the changes in one of them are related to the user access rate of the rest. Depending on the application our method can be used by itself, or as a supplementary to the existing methods, in order to increase the variety of the sources that the data is collected from, and improve the accuracy of the estimations.

In Section 2 we present the previous methods on web usage estimation and large scale DNS scans. In Section 3 we describe our methods. Section 3.2 presents methods to identify caches, and Section 3.3 describes our algorithm to crawl the caches. We discuss the estimation methods in Section 3.4, and give experimental results in Section 4. Finally, we conclude the paper in Section 5.

2. RELATED WORK

The DNS standard is defined in [13][14][15][16]. Location information in the DNS standard is defined in [7]. Kaminsky [11][12] reports nearly 2.5 million open access DNS caches in the world, based on scans in 2006. Grangeia [10] discusses possible attacks on DNS, including *DNS cache snooping* with recursive and non-recursive DNS queries. DNS cache snooping is also acknowledged as a low level security thread by the security community[2]. Our method is also based on non-recursively monitoring DNS servers for recent visits to websites.

Rajab *et al.* [18] use DNS cache probing among other methods to monitor some $800,000$ DNS caches to detect traces of bots trying to connect to their IRC servers. Felten *et al.* [9] discuss how an attacker can exploit DNS caching to obtain web usage history from an unsuspecting client. Their method assumes that the malicious code is embedded as a java script within attackers' website or sent to the victim through email.

Alexa [1] and Comscore [3] use toolbars to obtain anonymous data from their users in order to rank websites. Hitwise [4], Compete [2], and Quantcast [5] also do website ranking, but in addition to toolbars, these companies gather data from large ISPs. Our approach scales quite naturally in terms of geography, enabling us to aggregate usage statistics from a large region, or zoom into any area and easily get detailed usage statistics from that specific area. Also other methods collect data on all domains but have limited resolution in terms of geographic distribution of users, while our method allows us to get very detailed information about the distribution of accesses for a few domains of particular interest, by probing open DNS caches in an online manner. We limit the number of domains we monitor in order to prevent overuse of DNS caches, which is not an issue in previous work.

Ding *et al.* [8] use hyperlink structure in web pages to identify the geographical relevance of web pages, and adjust search engines to display local results first, if applicable. For example a national U.S. newspaper website is likely to have incoming links from all over U.S., while a regional newspaper only has links from its region. We can achieve the same effect by observing user accesses to the websites using DNS caches, such as national newspaper website will have visitors from all over the country, while a regional newspaper only attracts users from its region.

Poisson processes are widely used to model events if times between consecutive events are independent random variables, and the number of events in one interval is independent from the previous intervals. Once the events are modeled as a Poisson process, the unknown Poisson variable λ and its confidence interval is estimated within the defined confidence value by observing the events. Cho *et al.* [6] discuss different methods to estimate website update intervals. They assume that the websites are updated independently, and that updates can occur at any time. In their application the authors assume that the contents on the websites are updated as a Poisson process, and designed an estimator to guide their crawler, eventually improving the overall system performance and bandwidth usage. Estimating the Poisson variable λ is also discussed in [17] (page 311). The web sites are accessed by individual users at individual times, therefore it is logical to assume that the individual user visits could be modeled as Poisson events. This is the main reason we prefer to use Poisson estimators in our estimation procedure. Among the various estimators for Poisson, we adapt the estimator in [17] due to its tighter bounds in interval estimation.

3. DNS CACHE MONITORING

In this section, we present methods to use DNS caches to estimate usage profile for various domains. First, we present the problem setup, and describe DNS terminology used in this paper. Later, we describe our method to identify DNS caches, and present our algorithm to efficiently crawl the DNS caches for various domains. Finally, we present our algorithm to estimate the average user access rate for each domain, based on DNS cache observations.

3.1 Problem Setup

Given a set of open DNS caches that we have identified and a domain, the problem is to estimate the user access rate of the domain on these DNS caches by sending probing messages to the DNS caches and interpretting the responses. Each domain record is kept in the cache for a fixed time, defined as TTL. Typical cache times for domains differ from a minute to a day, and are obtained from each domain's authoritative server. Name resolution request to DNS caches are sent using recursive or non-recursive probes. Once a domain name is requested as a recursive probe, assuming it is not already in the cache, the DNS cache recursively finds the correct record and adds it to its cache. The record lives

[2]http://www.securityspace.com/smysecure/catid.html?id=12217

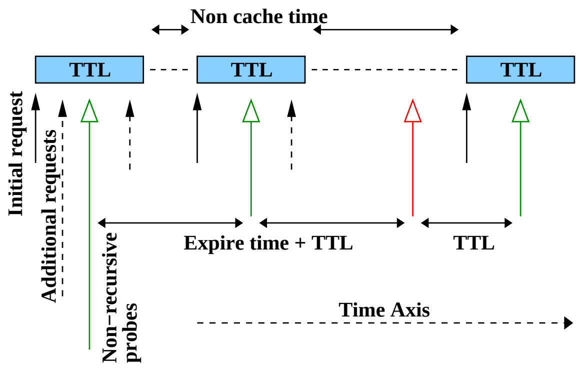


Figure 1: Different states of our non-recursive probes, and the user accesses on a typical DNS cache. The black arrows marked "Initial requests" and the dotted black arrows marked "Additional requests" are typical user interaction for the specified domain record. Green arrows represent our successful non-recursive probes, that we find the domain record in the cache. Similarly red arrow represents an unsuccessful probe. Initial requests are recursive user requests (from web browsers, etc.) that force the DNS cache to resolve and store the record. Once stored the record stays for TTL seconds, unaffected by additional user requests. We start crawling using non-recursive probes at an arbitrary time and adjust the frequency depending on the existence of the record in the cache, and record the sum of durations the domain was not in the cache (non-cache time) and the total number of successful non-recursive probes. We later use this data in our estimations.

in the cache up to TTL seconds, unaffected by further requests within that time period. During the time period the record is stored in the cache, all user requests to this record are answered using the DNS cache without consulting the authoritative or high level DNS servers. After TTL seconds, the record is removed from the cache until the next access time, when the whole process is repeated. Recursive probes are the typical way the user applications (e.g., web browsers) use for name resolution requests. As a result of the recursive probe, the cache state of the DNS server changes because of the newly added record. However in a non-recursive probe, the DNS cache only returns the name of the authoritative servers, but does not insert the record in its cache, thus keeps its cache unchanged by the probe. In both recursive and non-recursive probes, if the domain record is already in the cache, the number of seconds remaining for that record to expire is also returned, which we call expiration time. Simply by using a non-recursive probe, we learn if the domain record is already in the cache, and by comparing the expiration time with the TTL value we also learn the exact time the record was inserted into the cache, all without altering the cache of the DNS server. Altering the cache state is not desirable for two reasons; it forces our requested records to be included in the DNS cache, and because of this we cannot easily detect if the record was already in the cache or was added because of our interaction.

3.2 Identifying Caches

We categorize DNS servers that we identify on the web into two: (1) authoritative servers, which do not allow caching (most of the time) but answer to DNS queries related to their domains; (2) caches, which recursively resolve DNS queries for their users and cache the records for a certain period of time. In this paper, DNS caches are our main area of interest.

The first step of our method is to identify the DNS caches we would like to crawl. One easy and effective way is to scan an IP range for DNS caches [11] (e.g., IP range of U.S. universities, IP range of Switzerland, etc.). We do the scanning by sending valid DNS requests to each IP address, and interpreting the response. In order to separate caches from authoritative servers we set the recursive bit and ask for a specific domain (e.g. *poly.edu*) in our DNS request. As a result, we identify the DNS caches that actually *resolve* and *cache* the specific domain, apart from authoritative servers. We want to highlight here that recursive scanning is done only once in order to identify the DNS caches in an IP range. Once the caches are known, we only use non-recursive queries during crawling, thus keeping the cache state of DNS servers unchanged by our interaction.

In order to perform scanning, we developed our in-house software using the thread safe version of the libresolv C library, on a Linux box. Each thread sequentially reads the next available IP address from its pool, performs the scan operation, and writes the state of that IP address to an output file. In order to identify the DNS caches for our U.S. university crawls, we first selected 150 universities in IP ranges *128.x.x.x* and *129.x.x.x*, which leaves us with a total number of $150 * 2^{16} = 9,830,400$ IP addresses to scan. Each DNS query has an average size of 200 bytes, and setting the timeout for each query to one second we can scan one IP address per second per thread. With $1,200$ threads, we can scan the range in 2.27 hours, using 240 KB/s of the network resources.

3.3 Crawling Caches

The next step after identifying the DNS caches is to decide on a group of domain names to monitor. Crawls are performed by periodically probing the DNS caches for the existence of the domains we would like to monitor, as shown in Figure 1. The period depends on the TTL value and existence of the domain in the cache. We use only non-recursive probes to prevent altering the state of the DNS caches. We say that we perform an *exact crawl* if each event in the cache during some time interval t is guaranteed to be noticed by our crawler. In a *probabilistic crawl*, however, the probes can be random throughout the time interval, or the caches can be monitored for several shorter periods of time instead of the whole time interval. The number of probes affects the quality of the estimation. However, the number of probes to each DNS cache grows linearly with the number of domains we monitor, so due to resource limitations or to prevent overuse of DNS caches (especially for domains with small TTL values), probabilistic crawls might be more desirable. In this paper, we prefer exact crawls. The crawling continues periodically based on the TTL value of each domain as shown in Figure 1. In order to minimize the load on the DNS servers, our goal is to gather all the data using the minimum number of probes.

Assuming Q_{next} is the next scheduled query time, C is the time of the last query, TTL is the maximum time the record will remain in the cache before evicted, and R is the time in seconds the record will remain in the cache (expiration time) starting from C, we can give the following theorem:

Theorem 1. *In order to do an exact crawl with a minimum number of queries, we have to set the next query time Q_{next} as follows:*

$$Q_{next} = \begin{cases} C + TTL, & \text{if domain not in cache} \\ & \text{at time } C \\ C + R + TTL, & \text{if domain in cache} \\ & \text{at time } C \end{cases}$$

Proof. For the case in which the domain is not in cache at time C, we are guaranteed to catch the record in the cache if it was added (or catch the fact that it was not inserted) if we query again at time $C+TTL$. Therefore, the next query time becomes:

$$Q_{next} = C + TTL.$$

For the case when the domain is in the cache at time C, the record will stay in the cache for R seconds, and following the previous reasoning about the next query time, we can set the next query time as:

$$Q_{next} = C + R + TTL.$$

□

In case there are any delays expected in the network, Q_{next} can be set ϵ seconds earlier than the previously calculated value to prevent incorrect cache misses, where ϵ can be adjusted based on the delay in the network, but should be smaller than the TTL value. We also assume that the TTL values for domains are constants and do not change during the course of our scanning. Otherwise, we have to periodically verify the TTL value for each domain and adjust the query times accordingly.

Our crawler program is designed as a multi-threaded program. A single process reads the DNS cache IP addresses along with the domain names and their TTLs, and creates {*cache IP, domain, TTL*} tuples for each cache and domain name. Later, the process creates a priority queue and fills it with the tuples created. The priority queue orders the records based on next query time, and allows us to monitor domains with various TTL values. Once the priority queue is initialized with all the records, the process summons the threads to do the actual crawling as described in Theorem 1. The crawl results are stored on disk to be processed by our estimation algorithm, which we introduce in the next section.

3.4 Estimation

Crawling DNS caches gives us two types of valuable information: (1) the number of times a domain record is added to the cache, and (2) the duration in seconds the domain record is not in the cache. Combining the derived information with the TTL value of the domain, we can estimate the average access rate for a specific domain, on a specific DNS cache.

We assume the total number of user requests per second for a domain is a Poisson random variable χ with parameter λ. We monitor the DNS cache for a certain period of time T. T' is the total number of seconds the record for that domain is not cached (non-cache time), and h is the number of times that domain record is added to the cache. We use the estimator in [17] to estimate the Poisson parameter λ, and define our estimation $\bar{x}$ as the average number of estimated requests per second, calculated as:

$$\bar{x} = \frac{h}{T'}.$$

The intuition behind this estimator is that the average time a record stays outside cache after being evicted (average non-cache time) gives us the average access rate of the domain. The time interval between when the domain record is removed from the cache and the first user request for that same domain depends on the average access rate of the domain by the users. This time interval is our only chance to exactly measure the time between two consecutive user requests for that domain, since we cannot observe further requests once the domain record is cached. Thus the average non-cache time and the average user access rate are correlated, and can be better estimated by using a larger number of samples.

For a confidence value of 95%, and z_u^2 as a constant from statistics for 95% confidence level, the confidence interval for our estimation $\bar{x}$ becomes:

$$(\lambda - \bar{x})^2 = \frac{z_u^2}{T'}\lambda,$$

Rewriting the equation we get:

$$\lambda^2 - (2\bar{x} + \frac{z_u^2}{T'}) + \bar{x}^2 = 0,$$

The roots of the equation (λ_1 and λ_2) give the confidence interval for our estimation as:

$$\lambda_1 < \bar{x} < \lambda_2.$$

The accuracy of $\bar{x}$ and the tightness of the confidence interval change with the number of times the domain is observed in the cache (h), and the total duration of the observation. For a limited observation time the caching duration of the domain (TTL) also affects the observation accuracy since larger TTL values cause fewer cache observations (h).

Although our model initially assumes that each user visit to a website is a Poisson event, the joint behavior of users on a DNS cache is also a Poisson event. This comes from the fact that sum of Poisson events also follow Poisson distribution. Using this fact, we can generalize our estimation for multiple users and multiple DNS caches. Using our methods, we estimate the average access rate of a domain. Since we do not have a way to know exactly how many users are using each DNS cache, we can only talk about relative access rate of a domain. To overcome this limitation, we have to normalize our results using popular domains as our base since these domains are most likely accessed on every DNS cache. Search engines are good examples for such popular domains.

4. EXPERIMENTS

In this section, we present the experimental results of our algorithms both on simulated and real world environments. In Section 4.1 we describe the simulation environment used to estimate usage statistics, and present various experimental results based on our simulator. Experimenting with the simulator, we evaluate the effects of access rate, domain cache time (TTL), and observation duration on the final estimation accuracy. Later in Section 4.2 we use the experience from the simulator to crawl real world DNS caches that cover 150 U.S. universities spread out over five regions in the continental United States. We use this diverse sample to explore the effects of location on web access profiles, and identify some interesting trends.

4.1 Simulated Environment

Before working on real data, we first test our estimation algorithm on a simulated environment. The simulator mimics multiple user accesses to a single domain on a single DNS cache. As described in Section 3.4, we assume requests for a domain form a Poisson process with parameter λ, and we estimate this Poisson parameter using average non-cache time (T') and number of cache hits (h). Figure 2 (a) plots the error between λ and our estimate $\bar{x}$ for different DNS cache TTL values. The experiment is done for a fixed sized observation window of one day ($86,400$ seconds), and as seen from the figure, for smaller and larger values of λ the estimation is not accurate at all, but accuracy varies in between depending on the TTL value. The main reason for low accuracy for smaller λ values is simply that in one day we cannot see enough samples to estimate, and this gets worse for larger TTL values. Contrary, for λ values larger than 0.1, the observed record almost always sits in the DNS cache since the record is accessed on average every 10 seconds or less, and once accessed it sits in the cache for TTL seconds. For a good estimate, we have to see the record in the cache enough times, but we also have to observe some times where the record is not in the cache. From Figure 2 (a) we can conclude that limited observation time (e.g. one day) should be avoided especially for TTL values larger than one hour ($3,600$ seconds).

Once we examine the DNS cache for a fixed period of time (e.g. one day), we get different numbers of samples from domains with different TTL values. In order to more accurately evaluate our estimation method, and see if we can get accurate estimates even for larger TTL values, we design a new experiment to monitor until each domain is seen 100 times in the cache, and present the results in Figure 2 (b). As a consequence, the observation duration for each TTL differs, while the number of times the domain is seen in the cache is the same. As we see from the figure, the estimation algorithm gives acceptably accurate results if we can observe the domain enough times in the cache (increasing our sample size), independent of the domain TTL value.

We conduct another experiment to see how many samples we need from each TTL value to get less than 10% error on the access rate λ and our estimate $\bar{x}$. In order to increase the robustness of the experiment, we continuously probe the cache until the error is below 10% for three consecutive probes that we observe the domain in the cache, and report the number of probes conducted. Figure 2 (c) shows that we can get less than 10% error on each TTL value by getting at most 50 samples, which means observing the domain 50 times in the cache.

All the experiments up to now suggest that we can get good accuracy bounds by using our estimation method, and the accuracy of the estimation changes with the domain TTL value, user access rate for that domain, and our observation window size. For domains with larger TTL values or less frequently accessed domains it is better to do the observation based on the number of times we see the domain in cache, instead of a fixed time period.

4.2 Real-World DNS Crawls

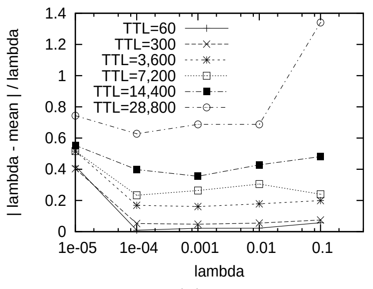
In this section, we present experiments done using real DNS caches from 150 U.S. universities within IP range *128.x.x.x* and *129.x.x.x*. The universities are used since it is more likely to find open DNS caches in universities, and they are easy to locate geographically. We use a freely available online service[3] to geocode the universities. We identified $4,818$ caches that responded to our initial scan. Although not all of these caches are expected to be accessible at all times during our crawls (some may have dynamic IP addresses and may be personal computers or laptops), an average of $3,000$ caches remained responsive during our crawls. In theory, DNS cache times can be quite large, say 24 hours or more, but in practice we observed that most of the popular domains use smaller cache times. The DNS cache times of the domains that we monitored in this paper vary from 300 to $7,200$ seconds. During our crawls, we showed utmost care to not overuse the DNS caches, and cause any inconvenience. In order to do so, we limit the number of domains that we monitor in each experiment, and avoid probing the same DNS caches frequently.

Before going into the details of the real-world experiments, we would like to highlight that we monitor the universities remotely, meaning that we do not have any insight about how many people are studying/working in each university, whether the DNS caches are official name servers or working on a student's laptop, and how many users each DNS cache has. In our case, knowing the exact number of users is not necessary, as we are more interested in comparing the relative access frequencies of different domains, and discovering usage trends rather than exact hit counts for each domain. We clearly do not have any prior information about the user profiles of these universities, and we can not know the exact website access profiles of these users. Our method gives accurate estimations based on the DNS caches that we monitor, but this is only a subset of the web accesses for the monitored domains, as we can only monitor a small subset of the U.S. web users. Even in the universities that we monitor, there may be other highly used DNS servers that we do not have access to.

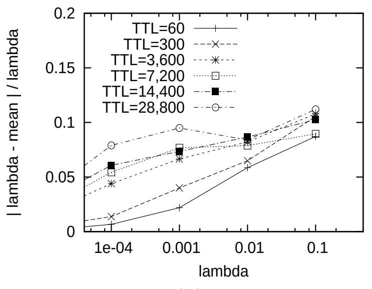
In order to compare popularities of domains between different universities, we introduce the notion of *Search Engine Units* to normalize access frequencies relative to the access rates of major search engines, since it can be argued that search is universally popular across all institutions. We use the collective access rate of three day crawls of *google.com* and *search.yahoo.com* as our base access rate.

In order to explore the effects of location on web usage

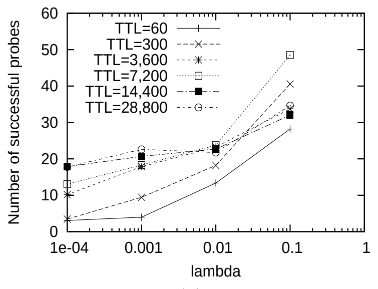
[3]http://geotool.servehttp.com



(a)



(b)



(c)

Figure 2: (a) Relative error of estimating λ, for different cache TTL times for a limited observation period of $86,400$ seconds (one day). The accuracy of the estimation varies with the TTL values of the domains, and user access rate. (b) Relative error of estimating λ, when the observation is not time bounded but continues until the domain is seen 100 times in cache for each TTL. (c) Number of successful probes required to get less than 10% error in our estimation for each TTL value. A Successful probe means we observe the domain record in the cache for that probe.

profiles we analyze the U.S. results in five regions (West, Southwest, Midwest, Southeast, and Northeast). The average access rates of individual universities are aggregated to find the total access rates of their regions. Using the regions we can display the effects of locality based on these regions. We crawled national and regional newspapers for a period of one weekday to observe the effects of location on web access profiles of different universities. Figure 3 plots the different access profiles for four different newspapers. As we see from the figure, the San Francisco Chronicle (sfgate.com) is more popular in the West, and the Boston Globe (boston.com) is mostly accessed in the Northeast. However, the Los Angeles Times (latimes.com) and the New York Times (nytimes.com) have a more general distribution. Figure 6 gives a more detailed view of the access rates of each newspaper for the same experiment. In this figure, we can observe the individual access rates per university. The results are color coded for easier presentation and colors of Red, Blue, Green and Yellow represent the popularity from higher to lower respectively. In Figure 6, we can see which specific areas within regions contribute to the access rates of the newspapers. For example in Figure 6 (b) we see that the popularity of the Los Angeles Times in West mostly comes from the areas around Los Angeles. Similarly in Figure 6 (d) we observe that the Boston Globe is not homogeneously accessed in the entire Northeast region, but the newspaper is only highly popular in a small area around Boston, and the popularity in the rest of the Northeast region is on par with the popularity in other regions of U.S. We also detected periodic accesses to all news sites that we monitored from a few universities. The exact same periodic access rate suggests that there is a machine involved in the process, possibly a news crawler. We omitted the crawlers that we detect from our results. The results for the San Francisco Chronicle in Figure 3 are a little odd, especially in the West. When we check these results, we observe that the San Francisco Chronicle access rates in some universities in West are quite higher than the combined search engine accesses, which boosts the *Search Engine Units*.

In Figure 7, we compare the user access rates of three popular online shopping sites: Amazon, eBay, and Barnes & Noble. We color coded the results based on which site is more popular than the rest in each particular university. Red color represents places where Amazon is more popular than the rest, while similarly yellow color represents eBay, and blue color represents Barnes & Noble. As we see in Figure 7 (a), Amazon and eBay are the more popular ones among the three, but it is hard to conclude whether there exist any spatial correlation within these results. From this figure, we can also detect some interesting outliers, such as Barnes & Noble's popularity in two universities in Texas. Figure 7 (b) shows how we can get detailed statistics from each single university.

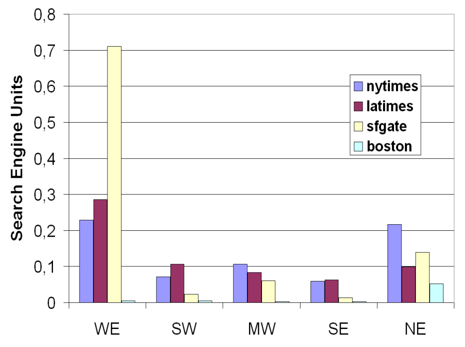
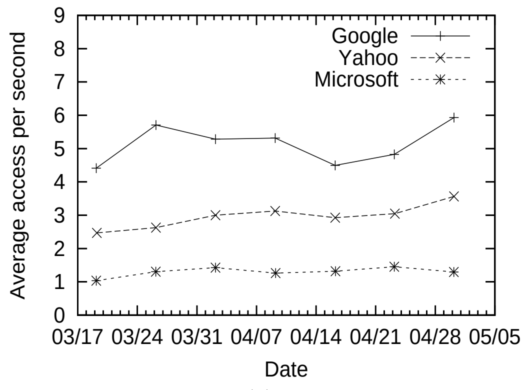


Figure 3: Newspaper access rates for US universities, normalized by the search engine access rates for each university. Figure plots the results based on five U.S. regions

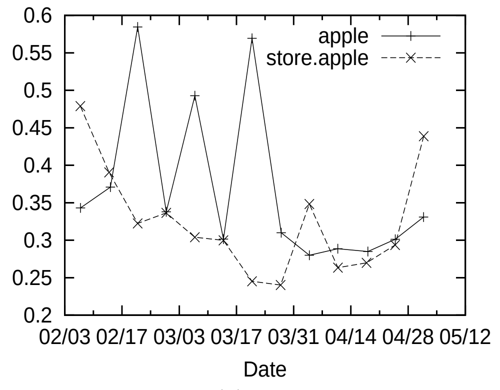
These figures are mere observations than conclusions as there are certain things that we can directly observe from the figures but cannot elaborate on the exact reasons. For example in Figure 6 (c), the popularity of San Francisco Chronicle in the West is expected, but the popularity of the same newspaper in certain locations in the Northeast region is interesting. Same goes for the popularity of Barnes & Noble in Texas (see Figure 7). As we do not have any internal data on the exact student profile of these universities, we cannot conclude why this is the case.

In Figure 4, we show a comparison of popular online mapping services. We observe that the Google Maps website is clearly more popular than the rest, followed by Yahoo Maps, Microsoft Local Live, and Mapquest. In this case location does not play much of a role on the ordering of these services.

We also monitored some domains for several days uninterrupted to observe the variations of web access rate of these



(a)



(b)

Figure 5: (a) Continues crawl of U.S. university DNS caches for various search engines, and (b) for apple.com and store.apple.com domains, in 2007.

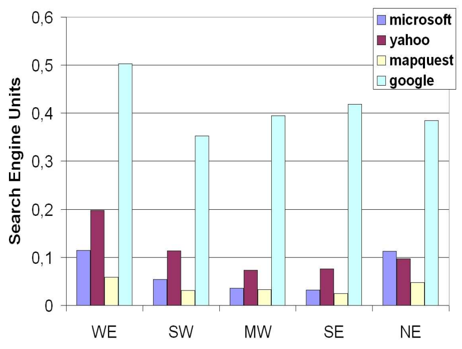


Figure 4: Online mapping service comparisons for U.S. universities, normalized by the search engine access rates for each university. The figure shows the results based on five regions in the U.S.

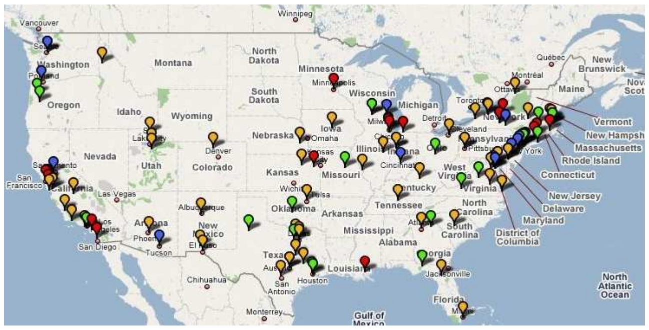
domains over time. Figure 5 (a) plots the total access rate of all U.S. universities for three popular search engines, and Figure 5 (b) for different Apple domains. The results given in this figure are not normalized by *Search Engine Units*, and are smoothed for each week. In all days, the order of the search engine accesses does not change, and Google is the most popular in search, followed by Yahoo Search and MSN Search (including Live Search). In Figure 5 (b) we observe sharp peaks in Apple's access rate. These days correspond to Mac OSX security patch distributions, and Apple TV shipment announcements, which is an interesting observation to us.

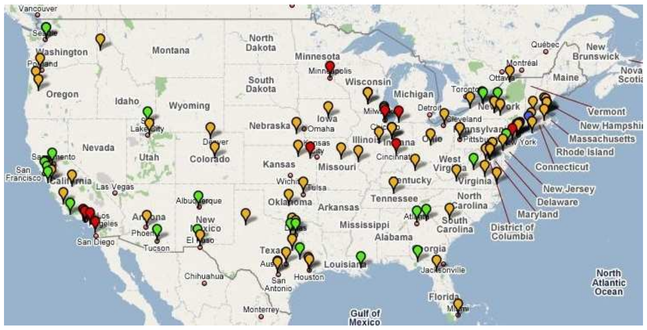
5. CONCLUSIONS AND FUTURE WORK

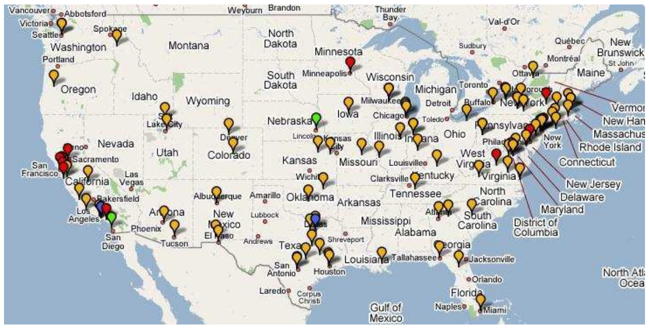
In this paper, we presented methods for estimating the usage rate of websites by monitoring DNS caches. We describe separate methods to identify the caches, crawl the caches for various domains, and finally estimate usage. We test our algorithms on a simulated environment, as well as on real world DNS caches from 150 U.S. universities. Our experiments show that we can estimate the usage of various domains, and we can observe various correlations of data based on geographical location and observation time. In future work, we would like to examine different estimators, and extend our DNS cache pool in order to perform more global scale crawls to make more detailed observations. We also would like to do more detailed observations on the outlier cases that we detected in our experiments. We are also interested in other geographical observations and possible applications of this technique.

6. REFERENCES

[1] http://www.alexa.com.
[2] http://www.compete.com.
[3] http://www.comscore.com.
[4] http://www.hitwise.com.
[5] http://www.quantcast.com.
[6] J. Cho and H. G. Molina. Estimating frequency of change. *ACM Transactions on Internet Technology (TOIT)*, 3:256–290, August 2003.
[7] C. Davis, P. Vixie, T. Goodwin, and I. Dickinson. [RFC 1876] A Means for Expressing Location Information in the Domain Name System, January 1996.
[8] J. Ding, L. Gravano, and N. Shivakumar. Computing geographical scopes of web resources. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB'00)*, pages 545–556, Cairo, Egypt, 2000.
[9] E. W. Felten and M. A. Schneider. Timing attacks on web privacy. In *ACM Conference on Computer and Communications Security (ICC'00)*, pages 25–32, Athens, Greece, 2000.
[10] L. Grangeia. DNS cache snooping: http://www.sysvalue.com/papers/dns-cache-snooping.
[11] D. Kaminsky. Black Ops Of TCP/IP 2005.5, New Explorations: Large Graphs, Larger Threats. http://www.doxpara.com/Black_Ops_Of_TCPIP_2005.ppt.
[12] D. Kaminsky. Explorations in namespace: white-hat hacking across the domain name system. *Commun. of the ACM*, 49(6):62–69, 2006.
[13] P. Mockapetris. [RFC 882] Domain Names, Concept and Facilities, November 1983.
[14] P. Mockapetris. [RFC 883] Domain Names, Implementation and Specification, November 1983.
[15] P. Mockapetris. [RFC 1034] Domain Names, Concept and Facilities, November 1987.
[16] P. Mockapetris. [RFC 1035] Domain Names, Implementation and Specifications, November 1987.
[17] A. Papoulis and S. U. Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 4th edition, International edition, 2002.
[18] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Internet Measurement Conference (IMC'06)*, pages 41–52, Rio de Janeiro, Brazil, 2006.


(a)


(b)


(c)

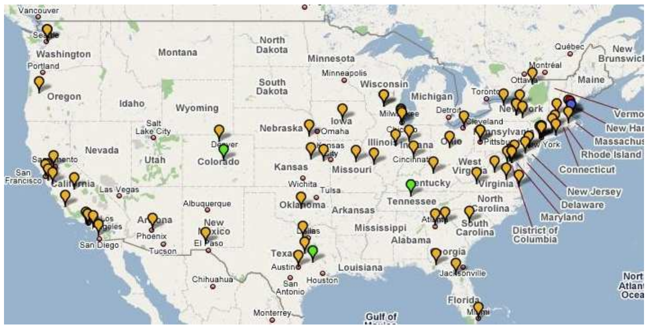
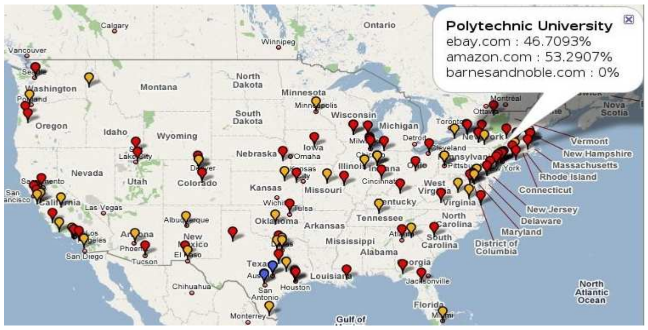
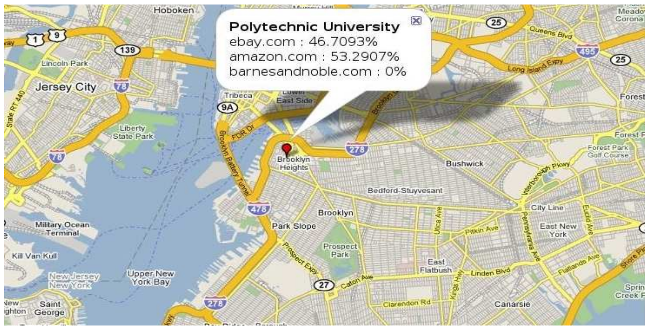

(d)

Figure 6: Detailed view of the results in Figure 3. The colors of Red, Blue, Green and Yellow represent the scale of access rates from higher to lower. The color scale is relative within each figure itself. (a) Access rate of the New York Times, generally distributed over the U.S. map. (b) Access rate of the Los Angeles Times, again generally distributed, with some concentration in the LA area. (c) Access rate of the San Francisco Chronicle: except a couple of places in the Northeast, most of the concentration is around Bay Area. (d) Access rate of the Boston Globe: lower access rates all over U.S. with high concentration around Boston area.



(a)



(b)

Figure 7: Comparison of online shopping sites. Red color represents universities where Amazon is more popular than the rest. Similarly yellow color represents eBay, and blue color represents Barnes & Noble. Using our method we can get an overview of the whole U.S. (a), or we can observe the results specific to one university (b).