Peeking Through The Cloud: DNS-based estimation and its applications

Moheeb Abu Rajab[1], Fabian Monrose[1], Andreas Terzis[1], and Niels Provos[2]

[1] Johns Hopkins University, Baltimore MD 21218, USA
moheeb,fabian,terzis@cs.jhu.edu,
[2] Google Inc., Mountain View, CA 94043, USA
niels@google.com

Abstract. Reliable network demographics are quickly becoming a much sought-after digital commodity. However, as the need for more refined Internet demographics has grown, so too has the tension between privacy and utility. Unfortunately, current techniques lean too much in favor of functional requirements over protecting the privacy of users. For example, the most prominent proposals for measuring the relative popularity of a website depend on the deployment of client-side measurement agents that are generally perceived as infringing on users' privacy, thereby limiting their wide scale adoption. Moreover, the client-side nature of these techniques also makes them susceptible to various manipulation tactics that undermine the integrity of their results. In this paper, we propose a new estimation technique that uses DNS cache probing to infer the density of clients accessing a given service. Compared to earlier techniques, our scheme is less invasive as it does not reveal user-specific traits, and is more robust against manipulation. We demonstrate the flexibility of our approach through two important security applications. First, we illustrate how our scheme can be used as a lightweight technique for measuring and verifying the relative popularity rank of different websites. Second, using data from several hundred botnets, we apply our technique to indirectly measure the infected population of this increasing Internet phenomenon.

Keywords: Client Density Estimation, Web-metering, Botnets, Network Security.

1 Introduction

Over the past few years, it has become increasingly important to garner reliable information about the demographics of the Internet and the myriad of services that it supports. For one, Internet businesses increasingly rely on such information to better customize their marketing campaigns. Advertisers, for example, make continual use of the relative popularity of websites and the demographics of their visitors to design and position their products and services in an effective manner. Similarly, security practitioners frequently use information about the population affected by security incidents to develop a better understanding of the characteristics and the scope of these attacks.

However, as the need for network demographics has increased, so too has the tension between utility and privacy. For instance, the most well-known schemes for measuring the relative popularity of websites (e.g., Alexa [38], ComScore [20] and NetRatings [26]) collect client-side data from deployments of measurement agents placed inside edge networks (e.g., using browser toolbars that record all URLs visited by the client). Generally speaking, the collected data is sanitized and used to produce aggregate statistics for the application in question. However, since the type of sanitization that is applied, as well as the specific data collected, are completely at the discretion of the data collector, large cross-sections of the population shy away from deploying these agents. Moreover, the mere client-side nature of these collection schemes opens the door to abuse (be it via click fraud [23] or other manipulation recipes [1]) that directly affect the integrity of the results. Recently, the prevalence of these fraudulent behaviors has raised so much doubts about the integrity and authenticity of these ranking measures that it captured the attention of the mainstream press (e.g., [22]).

Also of much interest lately is the question of how to reliably determine the infected population of an all too common security event, namely, botnets. Clearly, the size of the population affected by a particular security incident plays an important role in fully understanding its impact, as well as helps in prioritizing defense tactics from industry and practitioners alike. Unfortunately, although information on the scale and nature of a security event can be valuable for forensic and defenses purposes, network operators are usually reluctant to release such information as disclosure of (repeated) breaches can lead to loss of public confidence. Therefore, information on the spread of security events (worms, botnets, etc.) is normally collected at a global scale by dedicated measurement entities (e.g., CAIDA) using a combination of direct [28, 31] and indirect methods [3, 24, 31]. While these approaches have been widely successful, they are known to be vulnerable to various evasive tactics. For example, network monitors can easily be detected and evaded by active probing attacks [5, 29, 36]. Similarly, in the case of botnets, several practices complicate direct measurements as botmasters often suppress broadcast feedback, thereby making direct measurements infeasible even if the botnet has been infiltrated [30].

In what follows, we present a new technique for inferring the density of clients accessing a particular network service. Among a number of possible applications, our technique is directly applicable to both of the aforementioned problems. Specifically, we present an indirect estimation technique using DNS cache probing [14], and show how it can be used for website metering as well as for inferring the infected population of certain security events (e.g., botnets). In the former case, our evaluation shows that the technique is very accurate, and can serve as a standalone verification tool for determining the popularity rank of different websites. Compared to other approaches (e.g., Alexa [38]), our technique is less invasive as it does not require host-specific information. Moreover, as we discuss later, our technique is more robust under a threat model where the attacker is deliberately trying to inflate the popularity rank of her website.

In the latter case, we illustrate how our technique can be used to arrive at a better size estimate (than the notion of botnet *footprint* we suggested earlier [31]). We argue that this refinement is important as botnets continue to be one of the top Internet threats today [32, 37], and so more accurate size measurements have immediate benefit in assessing the monetary impact and damage they cause (e.g., via identity theft, DDoS attacks, etc.) [15]. While fine-grained estimates of botnet size still remain challenging [30], we believe this work offers a valuable step forward in that regard.

The remainder of the paper is organized as follows. In Sections 2 we illustrate our methodology and estimation techniques. In Section 3 we validate our approach through simulation and by comparison to an actual client count measured directly from our local network. In Section 4 we provide two real world applications that we believe aptly demonstrate the utility of our technique then we discuss some practical considerations in Section 5. In Section 6 we review related work, and conclude in Section 7.

2 Estimation Methodology

Growing security [18, 34] and privacy concerns raise significant challenges for the application of direct methods to obtain faithful counts of clients using a particular service, for example, by simply taking measurements from within network boundaries (e.g., using toolbars that monitor a users' browsing habits). Rightfully so, this unease calls for indirect counting techniques that limit the privacy risks with recording host-specific information. In this section, we describe our methodology for estimating the number of clients accessing a particular network service using a purely indirect technique.

Our approach exploits the fact that most network services (e.g., websites, botnet command and control servers) use DNS names to identify their servers. This, in turn, makes DNS resolution a pre-requisite step for any client connecting to that server. Simply speaking, we exploit this association to infer the number of clients requesting the resolution of the DNS name for the service of interest (e.g., `www.cnn.com`) from their local DNS resolver. Specifically, we use DNS cache probing to measure the evolution of that name in the resolver's cache and consequently derive an estimate of the number of clients accessing that service. Compared to direct methods, our technique is less intrusive as it does not reveal the specific identities of clients accessing the service of interest.

The technique itself is rather straightforward: for each DNS name of interest, S, we probe the cache(s) of the DNS resolver(s) for the network(s) of interest at regular intervals and examine the observed cache hits, if any, for S. For each cache probe, a cooperative resolver (i.e., a resolver that responds to DNS cache queries) will report a hit if S was in its cache, or a miss otherwise. In the former case, the resolver also reports the remaining time before S is flushed. While a cache hit only indicates that *at least one* client made a request for that entry, we can refine that estimate by sending appropriately-spaced probes that reveal the sequence of start and end times of S's entry in the resolver's cache. These times are a direct result of the combined queries from all clients that the resolver serves.

Figure 1 illustrates our estimation methodology. At a high level, one can envision the client population estimation as involving two processes: an input process representing the combined arrivals of DNS queries for the DNS entry S, from clients served by the same resolver, and an output process representing the refresh and expiry times of S that result from the input querying process. Our subsequent analysis relies on the simplifying assumption that the inter-arrival times of client requests in the input process can be modeled as a sequence of independent identically distributed random variables (IID's). Jung *et al.* [21] also used this assumption and showed that it does not introduce significant bias in the arrival model. We will return to this assumption, as well as the arrival model, in Section 2.2. Given this model, our goal is to estimate the number of clients n, requesting lookups for S from their common DNS resolver. We do so by estimating the aggregate DNS query rate λ, using the observed refresh and expiry times of the entry S from the output process. As we show later, the resulting estimate for λ leads to an estimate of the number of clients n. In what follows, we begin by describing our methodology for estimating the rate λ and then proceed to show how we can use this estimate to infer the number of clients n in Section 2.2.

2.1 Estimating the aggregate rate

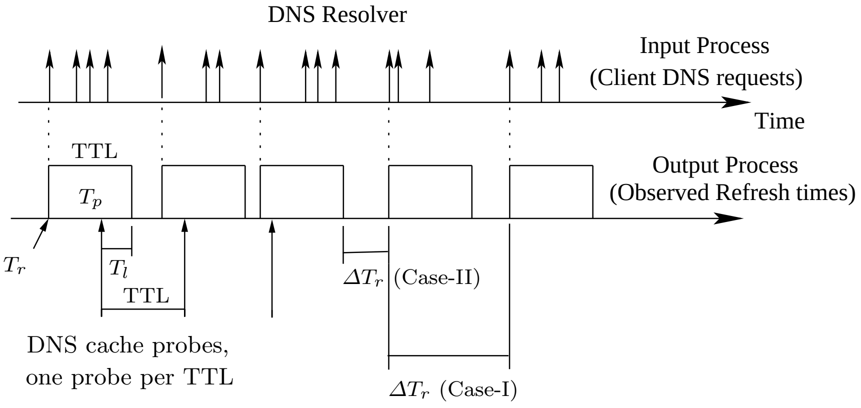


Fig. 1. Illustration of the estimation methodology.

For a DNS entry S, with a time to live (TTL), we estimate the aggregate rate λ, as follows: we probe the cache of the resolver of interest at a rate of one probe per TTL. As Figure 1 illustrates, the sequence of probes allows us to capture the start and end times of S in the resolver's cache. Recall that for a cache hit corresponding to a probe p at time T_p, the resolver returns the time T_l

until the cached entry expires. Given the TTL we can therefore infer the most recent start time (which we call the *refresh* time) T_r, as:

$$T_r = T_p - (TTL - T_l) \tag{1}$$

Then one way to estimate the average rate λ, is to compute the average time between consecutive refresh times $T_{r_1}, T_{r_2}, ..., T_{r_R}$ (Case I in Figure 1) from a sufficient number of refresh events R. Let ΔT_{r_i} be the time between consecutive refresh events of the entry S as observed from the output process, ($\Delta T_{r_i} = T_{r_i} - T_{r_{i-1}}$), then,

$$\lambda \approx \frac{R}{\sum_{i=1}^{R} \Delta T_{r_i}} \tag{2}$$

However, notice that this method is overly conservative since it assumes that no DNS queries arrive within the TTL of S in the resolver's cache. This is of course too restrictive and will lead to under-estimating the rate λ. Instead, we consider ΔT_{r_i} as the time between the expiry of the entry until its next refresh time (Case II in Figure 1),

$$\Delta T_{r_i} = T_{r_i} - T_{r_{i-1}} - TTL \tag{3}$$

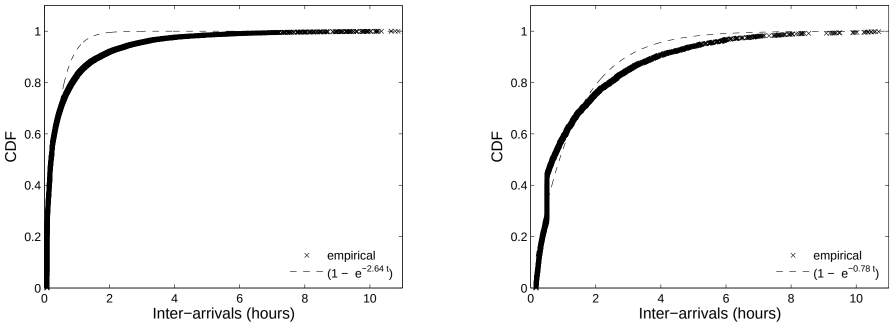
Notice that doing so makes the implicit assumption that the last DNS query in the input process took place slightly before the DNS entry expired. As we show later, this is not a significant issue and our technique still yields a fairly accurate estimate for practical TTL ranges.

Based on the newly calculated ΔT_{r_i}, we use Equation 2 to calculate the estimated rate λ from a sufficient number of refresh events R. To determine the required number of samples R, we apply the results of the central limit theorem [33]. For an acceptable error e, and confidence $z_{\alpha/2}$, we can calculate the sample size accordingly (see Appendix A for the detailed derivation). Finally, with the estimated λ at hand, we can infer the number of clients as a function of λ and the individual client request rate λ_c. In the next section we discuss how we estimate both λ_c and n.

2.2 DNS Request Arrival Model

In order to estimate the number of clients n, we need some knowledge of the DNS request arrival model. We derive this model by studying the distribution of the inter-arrival times of incoming DNS requests to a particular resolver. Specifically, we use a large dataset of over 320 million NetFlow records collected at the edge of a large campus network during a 24-hour period on 7/15/2007. We use this dataset to study the arrival models for two popular domain names, namely, `www.google.com` and `www.cnn.com` with TTLs of 5 and 10 mins, respectively. Assuming that each HTTP connection is preceded by a DNS request, we deduce these models by extracting the inter-arrivals of the start times of flows originating from individual hosts and destined to each one of these domains. Jung *et al.* [21] used a similar approach in their study on the effectiveness of DNS caching. Figure 2 shows the distribution of the inter-arrival times of requests to each name.

As the graphs show, the incoming client DNS request arrivals can be reasonably modeled by exponential random variables with different rates λ_c ($= 2.6$ queries/hour for `www.google.com` and 0.78 queries/ hour for `www.cnn.com`).



(a) `www.google.com` ($\lambda_c = 2.63$ *queries/hour*) (b) `www.cnn.com` ($\lambda_c = 0.78$ *queries/hour*)

Fig. 2. Cumulative Distribution Function (CDF) of the incoming client request inter-arrivals.

Following the assumption from Section 2, the sequence of IID exponential inter-arrivals from n clients (each with an input rate of λ_c) generates an output arrival process with Gamma distributed arrival times $Gamma(n, \lambda)$. Since n in our case is an integer value, then it follows from the gamma distribution [33] that the output process has exponentially distributed inter-arrivals with an aggregate mean rate of $\lambda = n\lambda_c$. We use this property to indirectly estimate n from the measured output process rate λ, where λ is estimated using R refresh events as illustrated in Section 2. Given λ, the expected number of clients, is $\left(n = \frac{\lambda}{\lambda_c}\right)$.

3 Validation

We first verify the accuracy of the proposed approach via simulation. Using the simulation parameters shown in Table 1, we evaluate the accuracy of our approach by measuring the estimation error of n for a wide combination of λ and TTL values. Figure 3 shows the estimation error. As the graph shows, our estimator is fairly accurate; the 95% confidence interval (i.e., within 2 standard deviations) of the mean estimation error remains within the bounds of the acceptable error set in the simulation.

3.1 In the Wild Evaluation

We further validate the effectiveness of our estimation technique by applying it to data collected from a real-world DNS probing experiment. In this experiment, we probe a local resolver serving a small department network for two popular DNS names, namely, `www.google.com` and `www.cnn.com`. For validation purposes, we enabled DNS logging on the local resolver so that all internal

Acceptable estimation error (e)	20%
Actual number of clients	100
Confidence	95%
Number of samples (R)	100

Table 1. Simulation parameters.

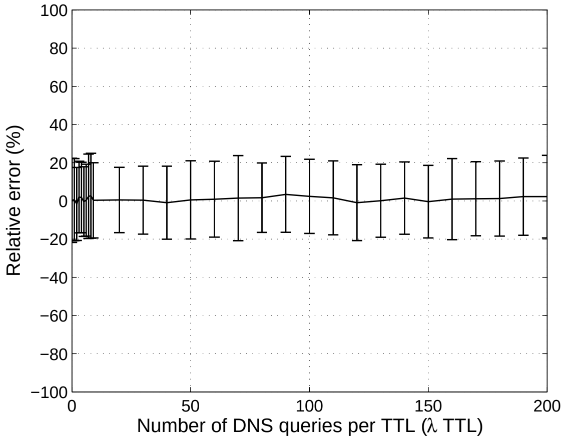


Fig. 3. Relative estimation error of the number of hosts n under different number of DNS queries per TTL ($\lambda\ TTL$).

DNS queries issued for either one of those names were recorded. We then extract the unique sources making queries for each name and use their count as a validation baseline for our estimate. Table 2 shows the parameters used in our experiment as well as the estimation results. The client query arrival rate, λ_c, is chosen based on the campus-wide trace discussed in Section 2.2.

As the table shows, the estimates are fairly accurate. For example, the estimated aggregate rate λ for `www.google.com`, from our cache probing, is 240 queries/hour. Dividing this estimate by the client query rate, $\lambda_c = 2.63$ query/hour estimated from the NetFlow dataset, yields an estimate of 93 clients accessed `www.google.com`. Similarly, our evaluation yields an estimate of 30 clients accessed `www.cnn.com`. Both estimates are within the bounds of the 20% error margin set in the experiment parameters. These results provide evidence on the viability of this estimation technique. We now turn to illustrate the flexibility of our approach through two important security applications: web-metering and botnet size measurement. We believe these two applications serve as good examples of the strength of our scheme.

	www.google.com	**www.cnn.com**
TTL	300 seconds	600 seconds
Client mean query rate (λ_c)	2.63 query/hour	0.78 query/hour
Cache probing rate	1 query every 5 mins.	1 query every 10 mins
Number of samples (R)	100	100
Acceptable estimation error	20%	20%
Actual number of clients (n)	**104**	**26**
Estimated mean aggregate rate (λ)	240 queries/hour	23 queries/hour
Estimated number of clients ($\hat{n}$)	**93**	**30**

Table 2. DNS cache probing experiment, parameters and results.

4 Applications

4.1 Estimating Website Popularity

As mentioned earlier, web metering (or popularity ranking) plays an important economic role in today's Internet. Popularity rank, for instance, is a key factor in deciding the marketing potential of a website. In particular, the higher a site's popularity rank, the more advertisers are willing to bid for advertising space on that site. Not surprisingly, because of the strong correlation between website popularity and monetary benefits, techniques for rank inflation are not uncommon [1, 2, 12], and so this problem has stirred much interest on the design of secure metering schemes (e.g., [16, 25]).

For the most part, web metering schemes attempt to address the problem of trust between advertisers and website owners by delegating the web metering task to a third party (e.g., Alexa [38], ComScore [20], NetRatings [26]) that monitors the interaction between clients and servers, and/or rely on cumbersome key agreement and distribution schemes [6, 16, 25]. In practice, the most well-known ranking services offer ranking for websites based on the number of visits they receive. These visits are measured from data collected from millions of users who willingly install measurement agents (e.g., Alexa toolbars [3]) on their machines. However, clearly such techniques raise security [18, 34] and privacy concerns as they reveal user-specific traits to the ranking service. Furthermore, the resilience and accuracy of these techniques has been recently brought into question [1, 22].

In what follows, we illustrate a simple, yet effective, web metering scheme that requires no client-side deployment. Additionally, our scheme is less intrusive as it does not breach individual users' privacy. The outcome of the proposed scheme is a list of website ranks measured from a completely different perspective. These ranks can be used as a stand-alone measure of the relative popularity of websites or to validate the results obtained from other ranking schemes.

Our technique is a direct application of our DNS-based estimation technique. That is, to measure the relative popularity rank for a set of websites, $\mathcal{B}$, we periodically probe the caches of a selected set of resolvers, $\mathcal{D}$, following the

[3] Whether or not these toolbars should be classified as *"Spyware"* seems to be a subject of much debate lately.

aforementioned approach (i.e., one probe every TTL epoch). Then to determine the relative rank of a website we simply measure the rate λ (as illustrated in Section 2) from the output of the probing process for each resolver in $\mathcal{D}$. The final rank K, is then expressed in terms of the average time it takes the web-site entry to be refreshed in the resolver cache after its last expiry (i.e., $1/\lambda$) across all resolvers in $\mathcal{D}$. Intuitively, DNS entries of websites with higher hit rates will be refreshed quicker than those with lower hit rates. To get the final website rank K we calculate the weighted average of the refresh times across all resolvers in $\mathcal{D}$,

$$K = \sum_{i \in \mathcal{D}} \frac{W_i}{\lambda_i} \tag{4}$$

where, W_i is the relative weight of each resolver in the final rank outcome. A number of criteria can be used to decide the relative weight for each resolver. For example, the weights can be decided based on the population demographics and target market of the advertising company, or from light-weight sampling of the IP-space (e.g., [9, 29]). In our case, we choose to apply a weight for each resolver based on the total client population served by that resolver. For simplicity, we infer this information from a dataset obtained from Google Inc. that contains a large list of resolvers and a coarse-grained estimate of the number of clients served by each resolver. For each resolver the weight W_i is then calculated as the number of clients served by resolver i divided by the total number of clients served by all resolvers in the sample $\mathcal{D}$.

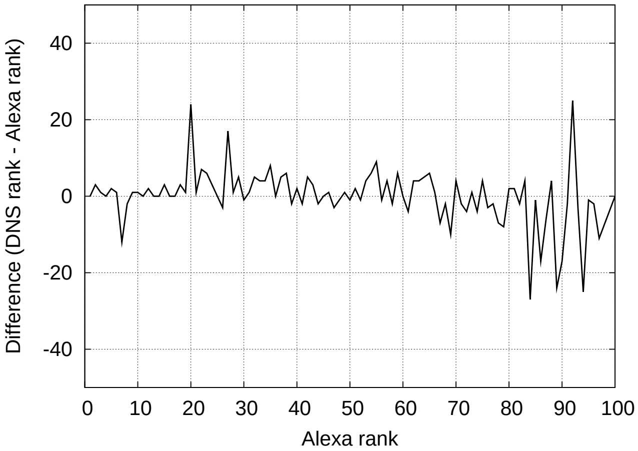


Fig. 4. DNS rank versus Alexa rank, for the top 100 websites according to Alexa ranking.

Case Study. We apply the above methodology to measure the relative popularity of the top 100 web-sites according to the ranking of `Alexa.com` [4]. For our target resolvers, we use a large list of 1.6 million resolvers obtained by collecting the Name Server (NS) records of a large list of crawled web URLs [27]. The resolvers list is first sanitized to extract the "cooperative resolvers" (resolvers that respond correctly to external cache queries). The sanitization phase involves sending two consecutive DNS queries to each resolver for a known DNS record. The first is a recursive query that forces the DNS server to fully resolve the query. The second query is sent with the recursion flag turned off to elicit a local reply from the resolver's cache. We compare the replies for consistency and also verify that the value of the TTL field in the second response is smaller than the one in the first response. After sanitization, a total of 768,000 resolvers were cooperative. As our target resolvers, we choose a smaller sample of resolvers from the sanitized list. We denote this sample $\mathcal{D}$. Notice that there are several ways to choose the sample $\mathcal{D}$. In our case, we first map the resolvers in the large list of 768,000 resolvers to their respective countries using the IP2Location database [13]. Our resolvers mapped to 189 countries. From each country we randomly choose up to k (=3) resolvers to form our final target list $\mathcal{D}$ of 495 resolvers [5]. We choose this particular methodology to serve our goal of ranking website according to their popularity from a global perspective. However, the selection criteria can be tuned to serve other ranking goals. For example, one could select all resolvers from a certain country to study web-site popularity with respect to users from that region. Investigating the selection of resolvers to serve such goals is outside the scope of this paper.

We probe each resolver in $\mathcal{D}$ for the top 100 websites from Alexa following the above methodology. Then we estimate λ_i for each resolver based on a sample of 50 probes and compute the final rank for each name using Equation 4. Figure 4 shows a comparison between our ranks and those of Alexa. As the graph shows, while both rankings show comparable results (with an average rank difference of 4.5), in some cases the ranks differ significantly. A closer look into some of the differing ranks reveal that they refer mostly to websites that have a country specific domain (e.g., `www.ebay.co.uk` had a rank difference of 22). Recall that we select our target resolvers from all-over the globe, hence this discrepancy is likely a consequence of the resolver selection criteria. In some other cases (e.g., `www.orkut.com` which has a rank difference of 12), the reason for the discrepancy in ranking is unclear. While it is difficult to argue for or against the accuracy of either ranking (without a true baseline) these results highlight the benefit of having metrics from different perspectives. This is important as the multiple measures can reveal inconsistencies in some ranks, and can be used further to produce new, and hopefully more robust, ranks based on a combination of different measures.

[4] We use the top 100 Alexa global ranks that are based on traffic statistics as of September, 2007.

[5] For countries containing ≤ 3 resolvers we choose all the available resolvers.

Resilience to Fraudulent Inflation

Click fraud. Click fraud schemes [23] have the dual effect of inflating the number of "click-throughs" on the Ads posted on a website and increasing the popularity of the website as the number of hits increases. While click-fraud may directly affect the ranks produced by direct counting schemes (e.g., those of Alexa), its effect on our ranks is more limited. For one, only those clicks originating from hosts served by resolvers in our randomly chosen sample, $\mathcal{D}$, may influence the ranking. More importantly, to influence the outcome of the probing process, these clicks need to persist over a long period of time in order to significantly change the average refresh rate.

Direct manipulation attacks. In light of our technique, one might attempt to inflate the popularity of a website by polluting the caches of the resolvers that we probe. Cache pollution is possible if the resolver allows recursive external DNS queries. In this case, the attacker may send a sequence of synchronized DNS queries for the service name of interest—spaced by the TTL—so that the DNS entry is refreshed immediately after its expiry. Consequently, our probing process will falsely yield a high refresh rate for the target resolver. However, for this attack to be effective, the attacker must target enough resolvers from $\mathcal{D}$ to influence the final website popularity rank. To mitigate such attacks, our sample of resolvers is selected at random, by region, and refreshed periodically, thereby making pollution attacks more difficult (though not infeasible) to perpetrate.

4.2 Estimating Botnet Size

Another compelling use of our technique is that of estimating the size of a botnet. Botnets are networks of compromised hosts, called bots, under the control of human operators, referred to as botmasters. Botnets are primarily used for various types of malicious activities, including denial of service attacks, click fraud [12], software piracy, and spam. While botnets have only recently attracted the attention of the research community, several works on the topic have already emerged ([10, 11, 19, 28, 31]). In particular, several studies have attempted to address the specific question of botnet size and the subtleties involved in size estimates. For example, Dagon *et al.* used DNS redirection to divert bot connections to a darknet in order to directly count the number of bots [11]. While effective, their approach requires coordination with DNS authorities in order to perform the redirection. Other studies used botnet infiltration to directly count the number of bot ID's observed on the botnet command and control channel [17, 31]. Unfortunately, botmasters are increasingly suppressing bot information on the command and control channel, thereby hindering the effectiveness of these techniques.

As a remedy, we proposed [31] a technique for estimating botnet sizes using DNS cache probing. Similar to this work, botnet sizes were measured by probing the caches of a large set of DNS resolvers for the DNS name of the botnet server. The total number of DNS resolvers returning a cache hit for the name in question provided a botnet's so-called "DNS footprint". However, this footprint is, *at best*, a lower bound of the true infection size because that approach does not provide any indication on how many infected bots reside behind a domain where a hit was returned.

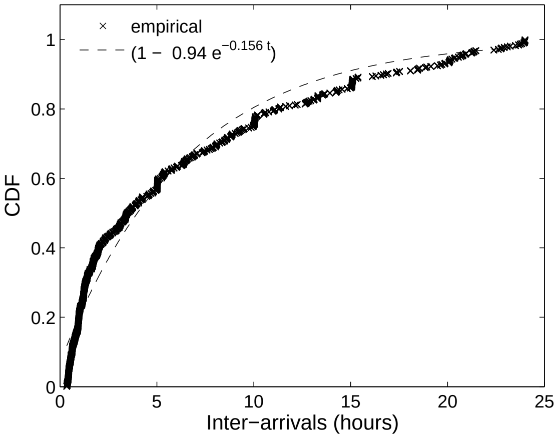


Fig. 5. CDF of Individual bot join inter-arrival times (based on data from 470 botnets).

In what follows, we illustrate how one can use the approach suggested in this paper to provide a better estimate of a botnet's population. To do so, we first examine the distribution of bot request inter-arrivals. We derive this distribution by studying the bot[6] join times extracted from a dataset containing the activity logs of 470 infiltrated IRC botnets [31]. Figure 5 shows the distribution of bot join inter-arrival times over a period of more than 9 months. As the graph shows, bot inter-arrival times can be approximated by an exponential distribution with an average rate of $\lambda_c = 0.156$ (i.e., an average of one connection every 6.4 hours).

Given knowledge of the distribution of bot request inter-arrivals, we now apply our estimation technique. In order to have a baseline for validation, we only choose botnets whose member counts are sufficiently large and can be calculated directly from the IRC traces. This yields two botnets, with member counts of 12,700 and 10,690 respectively. As our target resolvers, we use the large list of 768,000 cooperative resolvers from Section 4.1. Following the methodology from Section 2, for each resolver, we send a sequence of cache probes spaced by the TTL for the botnet server name in question then we estimate the botnet population by calculating the sum of the estimated number of bots n served by

[6] In our analysis we assume that a bot IP address is a sufficient measure of bot uniqueness. To account for the effect of DHCP we only consider join inter-arrivals that are $\leq$ 24 hours. We also exclude bot joins resulting from clone attacks as well as any join with no associated quit message.

each resolver. Table 3 summarizes the parameters of the probing experiment and the results of our estimation.

	Botnet I	**Botnet II**
TTL	15 mins	30 mins
Client mean query rate (λ)	0.156 query/hour	0.156 query/hour
Cache probing rate	1 query every 15 mins.	1 query every 30 mins.
Number of DNS resolvers	768,000	768,000
Number of samples (R) per resolver	100	100
Population size (from IRC logs)	**12,700**	**10,690**
Measured DNS footprint	**1,700**	**1,452**
Estimated population	**8,400**	**6,350**

Table 3. DNS cache probing experiment, estimating botnet sizes.

The probing experiment shows that the botnets in question have DNS footprints [31] of 1,700 and 1,452, respectively. For each resolver in the footprint, we extracted all refresh times for both botnet server names. We then applied our estimator from Section 2 to derive the size of the infected population of each botnet. Our estimation results show that the sizes of the infected population were about 8,500 and 6,350 bots, respectively. Clearly, the population estimates derived from our analysis are much closer to the actual population sizes compared to the more coarse-grained DNS footprints — that imply sizes of only 1,700 and 1,452 bots, respectively. That is equivalent to more than a *three fold* improvement in accuracy over the DNS footprint estimate.

The observant reader would note that the error margin from the actual bot count is larger than that in Section 3. The degradation in accuracy is due to the fact that our list of target DNS resolvers only covers a subset of all DNS resolvers in the Internet. Hence, a more comprehensive list of servers would enhance the estimation accuracy. Additionally, botnet size instability (be it due to bot migration or churn [31]) also contributes to this effect. Nonetheless, we believe our result shows the utility of this estimation technique in assessing a botnet's size when it is not possible to make such measurements directly even after the botnet has been infiltrated.

5 Practical Considerations

Notice that our probing mechanism requires cooperative resolvers that respond correctly to external cache probes. The sanitization step in Section 4.1 showed that roughly half of the resolvers in our list did not respond to external cache queries. While this is not a hindrance for some applications (e.g., for web-metering we only require a small sample of resolvers), having a large set of resolvers will improve the accuracy of other applications (e.g., botnet size estimation). An alternative probing model that can overcome this limitation would be to deploy a set of distributed DNS probing sensors inside the boundaries of large networks (e.g., within the boundaries of ISPs). The internal sensors will be able to query the caches of their respective resolvers and give an estimate for the client density in the same network.

Another noteworthy point of discussion is the practical impact of the TTL interval. Our analysis shows that the change in the TTL interval length does not significantly affect the accuracy of our estimator. However, in practice, the value of the TTL has an impact on the estimation speed and the overhead associated with the probing process. Recall that we probe each DNS name at a rate of one query per TTL. For large TTL values (e.g., on the order of one day), our probing scheme will require a long time to collect enough samples in order to reliably estimate λ. Luckily, major websites mostly use short TTL values for the purposes of load balancing [35]. Figure 6 illustrates the distribution of the TTL length for the top 100 websites in Alexa's ranking. As the graph shows, the majority of the TTLs are relatively short (about 85% of the TTLs are less than one hour). Likewise, our ealier work showed that a significant portion of the DNS names used by botmasters have short TTLs [31]. In many cases, these DNS names are served by dynamic DNS providers that intentionally use shorter TTLs to accommodate for frequent IP address changes of the subscribing servers. Finally, we note that one way to accommodate for large TTLs is to compute a running estimate of λ and keep updating the estimate as more samples are collected.

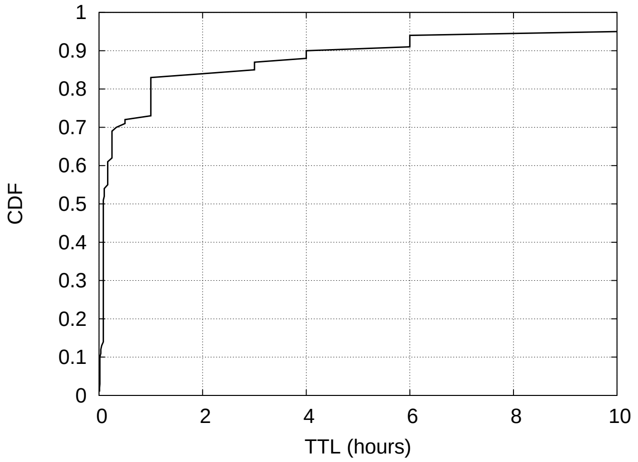


Fig. 6. CDF of the TTL intervals for the top 100 websites according to Alexa. The majority of websites use short TTLs with about 85% of the TTLs being less than one hour.

6 Other Related Work

The general problem of inferring the size of various client populations on the Internet has received considerable attention over the last few years. For the most part, the proposed techniques share the characteristic of attempting to estimate the size of a population in the absence of information from within the networks' edge. Generally speaking, these techniques differ in the inference mechanisms they use to derive the different population estimates. Additionally, each scheme is normally tailored to meet a specific goal for a specific context.

For example, Bellovin proposed a technique for estimating the number of hosts behind a Network Address Translation (NAT) device [4]. His technique is based on observing the evolution of the value of the identity field in the outgoing IP datagrams. More recently, Casado *et al.* used the number of scans received by strategically placed darknets to infer the percentage of Code Red II-infected hosts that resided behind NAT devices [8]. Additionally, Casado and Friedman proposed techniques based on active web content to estimate the number of hosts located behind a large number of NAT devices and web proxy servers [7]. Our work is different both in scope and technique. In our case, the goal is to estimate the density of clients accessing the same Internet service using DNS cache probing.

The strength of our approach is demonstrated by its utility for a wide range of applications, two of which are presented in Section 4. The web-metering problem has been the subject of a number of earlier research proposals (*e.g.*, [6, 16, 25]). At a high level, these schemes use cryptographic primitives to design web-metering schemes that are resilient to click inflation attacks (*e.g.,* [1, 2, 12]). These approaches, however, are resource intensive and require sophisticated key agreement and key distribution schemes. By contrast, our scheme is relatively straightforward and requires no client side deployment.

Population estimation techniques have also been used to estimate the size of infections caused by malware spreading. For example, Dagon *et al.* used DNS redirection to measure the number of hosts connecting to IRC servers associated with botnet C&C channels [11]. More recently, we used both direct and indirect methods to better understand the spread of botnets in the wild, and how to characterize their behavior [30, 31]. While these later works also use DNS probes, they are different from the approach in this paper in an important way: specifically, while our earlier work provides a course-grain estimate, we refine that approach to provide a technique for estimating (with reasonable approximation error) the number of infected hosts within these domains.

7 Conclusion

In this paper, we provide a new technique for estimating an important class of Internet demographics, specifically, the client population density of a given service. We demonstrate the utility of our approach through two applications that we argue are of much interest to the security and network community at large: verifying the popularity rank of a website and estimating the size of a botnet infection. Compared to earlier techniques, our popularity ranking scheme is easier to deploy, offers increased resilience to fraudulent manipulation, and is less intrusive as it does not reveal user-specific traits to the ranking service. In the second case, we provide a refined technique for estimating botnet size. We argue that since the issue of size plays an important role in assessing the monetary cost and damage caused by botnets, improvements in accuracy in estimating their size is of immediate benefit. In short, our approach yields a three-fold improvement in accuracy compared to the best previously known technique. We believe these results aptly demonstrate the utility of our approach.

Acknowledgments

This work is supported in part by National Science Foundation grant CNS-0627611. We thank Angelos Keromytis, Sal Stolfo, Angelos Stavrou and Joel Rosenblatt for providing access to the anonymized NetFlow records used in our analysis. We also extend our gratitude to the reviewers for their insightful comments and feedback.

References

1. 20 Quick Ways to Increase Your Alexa Rank. See `http://www.doshdosh.com/20-quick-ways-to-increase-your-alexa-rank`, 2007.
2. Vinod Anupam, Alain Mayer, Kobbi Nissim, Benny Pinkas, and Michael K. Reiter. On the security of pay-per-click and other web advertising schemes. In *WWW '99: Proceeding of the eighth international conference on World Wide Web*, pages 1091–1100, New York, NY, USA, 1999. Elsevier North-Holland, Inc.
3. Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, and David Watson. Internet motion sensor: A distributed blackhole monitoring system. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS)*, 2005.
4. Steven M. Bellovin. A Technique for Counting NATted Hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment (IMW)*, pages 267–272, 2002.
5. John Bethencourt, Jason Franklin, , and Mary Vernon. Mapping Internet Sensors with Probe Response Attacks. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 193–212, August 2005.
6. Carlo Blundo and Stelvio Cimato. Sawm: a tool for secure and authenticated web metering. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 641–648, New York, NY, USA, 2002. ACM Press.
7. Martin Casado and Michael Freedman. Peering through the shroud: The effect of edge opacity on IP-based client authentication. In *Proceedings of* 4^{th} *USENIX Symposium on Networked Systems Design and Implementation (NDSI)*, April 2007.
8. Martin Casado, Tal Garfinkel, Weidong Cui, Vern Paxson, and Stefan Savage. Opportunistic Measurement: Extracting Insight from Spurious Traffic. In *Proceedings of the* 4^{th} *ACM Workshop on Hot Topics in Networks (HotNets-IV), College Park, MD*, November 2005.
9. Zesheng Chen and Chuanyi Ji. A Self-Learning Worm Using Importance Scanning. In *Proceedings of ACM Workshop On Rapid Malcode (WORM)*, November 2005.

10. Evan Cooke, Farnam Jahanian, and Danny McPherson. The Zombie Roundup: Understanding, Detecting, and Disturbing Botnets. In *Proceedings of the first Workshop on Steps to Reducing Unwanted Traffic on the Internet*, July 2005.
11. David Dagon, Cliff Zou, and Wenke Lee. Modeling Botnet Propagation Using Time Zones. In *Proceedings of the* 13^{th} *Network and Distributed System Security Symposium NDSS*, February 2006.
12. Neil Daswani, Michael Stoppelman, the Google Click Quality, and Security Teams. The Anatomy of Clickbot.A. In *Proceedings of the first USENIX workshop on hot topics in Botnets (HotBots'07).*, April 2007.
13. IP2Location Database. See `http://www.ip2location.com/`.
14. DNS Cache Snooping or Snooping the Cache for Fun and Profit. Available at `http://www.sysvalue.com/papers/DNS-Cache-Snooping/files/DNS_Cache_Snooping_1.1.pdf`.
15. Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 375–388, New York, NY, USA, 2007. ACM.
16. Matthew K. Franklin and Dahlia Malkhi. Auditable metering with lightweight security. In *Financial Cryptography*, pages 151–160, 1997.
17. Felix Freiling, Thorsten Holz, and Georg Wicherski. Botnet Tracking: Exploring a root-cause methodology to prevent denial-of-service attaks. In *Proceedings of* 10^{th} *European Symposium on Research in Computer Security, ESORICS*, September 2005.
18. Exploiting the Google toolbar. GreyMagic Security Advisory. See `http://www.greymagic.com/security/advisories/gm001-mc/`.
19. Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In *Proceedings of the* 16^{th} *USENIX Security Symposium.*, pages 167–182, August 2007.
20. ComScore Inc. See `http://www.comscore.com`.
21. Jaeyeon Jung, Arther Burger, and Hari Balakrishnan. Modeling TTL-based Internet Caches. In *Proceedings of IEEE INFOCOMM*, 2003.
22. How Many Site Hits? Depends on Who's Counting. New York Times article. Louis Story. See `http://www.nytimes.com/2007/10/22/technology/22click.html?_r=3&pagewanted=1&ref=technology&oref=slogin`, 2007.
23. Ahmed Metwally, Divyakant Agrawal, Amr El Abbad, and Qi Zheng. On hit inflation techniques and detection in streams of web advertising networks. In *ICDCS '07: Proceedings of the 27th International Conference on Distributed Computing Systems*, page 52, Washington, DC, USA, 2007. IEEE Computer Society.
24. David Moore. Network Telescopes: Observing Small or Distant Security Events. In 11^{th} *USENIX Security Symposium, Invited Talk*, August 2002.
25. Moni Naor and Benny Pinkas. Secure and efficient metering. *Lecture Notes in Computer Science*, 1403:576–591, 1998.
26. Nielsen/NetRatings. See `http://www.nielsen-netrating.com`.
27. Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. What's New on the Web? The Evolution of the Web from a Search Engine Perspective. In *Proceedings of the* 13^{th} *International World Wide Web (WWW) Conference*, pages 1–12, 2004.
28. Honeynet Project and Research Alliance. Know your enemy: Tracking Botnets, March 2005. See `http://www.honeynet.org/papers/bots/`.

29. Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis. Fast and Evasive Attacks: Highlighting the challenges ahead. In *Proceedings of the* 9^{th} *International Symposium On Recent Advances In Intrusion Detection (RAID)*, pages 206–225, September 2006.
30. Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Adreas Terzis. My Botnet is Bigger than Yours (Maybe, better than yours): Why Size estimates remain challenging. In *Proceedings of the first USENIX workshop on hot topics in Botnets (HotBots'07)*, April 2007.
31. Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC)*, pages 41–52, Oct., 2006.
32. FBI Botnet Cyber Crime Report. See `http://www.fbi.gov/pressrel/pressrel07/botnet061307.htm`, June 2007.
33. Sheldon M. Ross. *Introduction to Probability Models.* Academic Press, 1993.
34. Unpatched Google Toolbar Flaw Presents ID Theft Risk. Ryan Naraine. See `http://www.eweek.com/c/a/Security/Unpatched-Google-Toolbar-Flaw-Presents-ID-Theft-Risk/`.
35. Anees Shaikh, Renu Tewari, and Mukesh Agrawal. On the Effectiveness of DNS-based Server Selection. In *Proceedings of IEEE INFOCOM 2001*, volume 3, pages 1801–1810, 2001.
36. Yoichi Shinoda, Ko Ikai, and Motomu Itoh. Vulnerabilities of Passive Internet Threat Monitors. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 209–224, August 2005.
37. FBI Computer Crime Survey. See `http://www.fbi.gov/page2/jan06/computer_crime_survey011806.htm`, 2006.
38. Alexa: the web information company. See `http://www.alexa.com`.

Appendices

A Deriving the required number of refresh events (R)

It is known that the average rate λ measured from multiple independent samples is normally distributed around the actual mean:

$$f(\lambda) = N\left(\lambda, \frac{\lambda}{R}\right). \tag{5}$$

For an acceptable estimation error e and a confidence $z_{\alpha/2}$ the number of samples R required is [33]:

$$R = \left(\frac{z_{\alpha/2}\ \sigma}{e}\right)^2. \tag{6}$$

where σ is the standard deviation of the measured mean rate, λ, which can be determined from a smaller pilot sample of DNS refresh events.