Estimating the Number of Users behind IP Addresses for Combating Abusive Traffic

Ahmed Metwally and Matt Paduano
Google Inc
1600 Amphitheatre Pkwy, Mountain View, CA 94043 USA
{metwally, mattp}@google.com

ABSTRACT

This paper addresses estimating the number of the users of a specific application behind IP addresses (IPs). This problem is central to combating abusive traffic, such as DDoS attacks, ad click fraud and email spam. We share our experience building a general framework at Google for estimating the number of users behind IPs, called hereinafter the sizes of the IPs. The primary goal of this framework is combating abusive traffic without violating the user privacy. The estimation techniques produce statistically sound estimates of sizes relying solely on passively mining aggregated application log data, without probing machines or deploying active content like Java applets. This paper also explores using the estimated sizes to detect and filter abusive traffic. The proposed framework was used to build and deploy an ad click fraud filter at Google. The first 50M clicks tagged by the filter had a significant recall of all tagged clicks, and their false positive rate was below 1.4%. For the sake of comparison, we simulated a naïve IP-based filter that does not consider the sizes of the IPs. To reach a comparable recall, the naïve filter's false positive rate was 37% due to aggressive tagging.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications—*Data mining*; C.2.0 [**Computer Communication Networks**]: General—*security and protection*

General Terms

Algorithms, Experimentation, Measurement, Security

Keywords

IP Size Estimation, Abusive Traffic Filtering, Advertisement Click Fraud, Real Data Experiments

1. INTRODUCTION

Simple conventional mechanisms for abuse detection that rely on source IPs set a limit, i.e., filtering threshold, on the IP activity within a time period. Once the limit is reached by an IP, either the IP traffic gets filtered for the rest of that time period, or the IP gets blacklisted for several consecutive periods. These techniques typically set the same threshold for all IPs. Setting an aggressive threshold yields a high false positive rate since some IPs have numerous users behind them and are hence expected to send relatively large traffic volumes. Setting a conservative threshold yields a high false negative rate, since the threshold becomes ineffective for distributed attacks where IPs send relatively little traffic. This work tailors the thresholds to the sizes of the IPs. It proposes a new framework for timely estimation of the number of users behind IPs with high enough accuracy to reduce false positives and with high enough coverage in the IP space to reduce false negatives.

We define the sizes of the IPs based on two dimensions: application and time. Each application has a specific size associated with each IP depending on the number of human users using this application. For Google, the query size of an IP is the number of human users querying the search engine, which may differ from the number of users clicking ads or the number of users sending emails to the Google email service. Thus, sizes should be estimated using the log files of the application whose activity is subject to estimation.

The other dimension for defining the sizes is time. The number of human users behind an IP changes over time, like when the IP observes a flash crowd, i.e., an unexpected surge in usage, or gets reassigned to households and/or companies. The size estimates should be issued frequently enough to cope with the frequent size changes. This calls for a short estimation time period. Conversely, the estimation period should be long enough to yield enough IP coverage, as well as enough traffic per IP to produce statistically sound sizes.

Estimation Challenges and Methodology

While estimating the sizes of individual IPs has ramifications on the security field, the primary concern is violating the user privacy. To preserve the user privacy, the proposed techniques estimate sizes of IPs using the application-level log files. First, the application users are assumed to be only temporarily identified, e.g., with cookie IDs in the case of HTTP-based log files. Thus, no Personally Identifiable Information, such as the name or the email address, is revealed. Second, no individual machines are tracked. Third, the framework uses application log data aggregated at the IP-level. Over 30% of dynamic IPs are reassigned every one to three days [19], and thus an IP is considered a temporary identification of a user. In addition, the majority of the

KDD'11, August 21–24, 2011, San Diego, California, USA.

users share IPs. This is illustrated by Fig. 1 that shows the distribution of 10M random IPs (from Google ad click log files) shared by 26.9M total estimated ad users.

Figure 1: The estimated sizes of 10M random IPs.

Estimating sizes from the log files is not straightforward. Naïve counting of distinct user identifications, e.g., cookie IDs or "user agents" (UAs), per IP fails to accurately estimate sizes. Corporate NAT devices usually have the same UA on all hosts. Similarly, an Internet cafe host is used by several users sharing the same user ID. Meanwhile, small IPs can masquerade as large IPs by clearing or farming cookies, and overwriting UAs in HTTP requests. Therefore, estimating sizes by distinct counting of cookies and UAs may result in over-estimation or under-estimation. Filtering traffic based on these inaccurate sizes yields high false negatives and high false positives, respectively.

Instead, this paper proposes using the log files to build statistical models that are later used for estimating sizes. However, this approach poses some challenges.

1. The log files do not contain only legitimate traffic. The existence of abusive traffic entries in these files degrades the quality of the models and the estimated sizes. To avoid such quality degradation, the models should be built only from the traffic of the trusted users[1]. This introduces a sampling bias in the traffic used to build the models. To mitigate this bias later in the estimation phase, only the trusted traffic of each IP[2] during a period, p, is used to estimate its size for p (§ 2.).
2. The sizes of the IPs change due to legitimate reasons, such as reassignments, flash crowds and business-week cycles. For an estimation period, p, the log files cannot be finalized before the end of p. They are then analyzed to produce estimates after each IP has already made its activities during p. Hence, estimated sizes are always lagging behind real-time sizes. Meanwhile, real-time abuse detection needs the estimates when p begins. This lag reduces the filtering accuracy when an IP legitimately changes size (§ 4).

[1]Trusted users, identified by cookie IDs for example, are typically those whose activity was rarely tagged as abusive by any existing filter. If this is the only deployed filter, trusted users can be alternatively defined as those with some signature of good traffic, where the definition of good traffic is application-dependent. For combating ad click fraud, trusted cookies can be defined as those with a relatively high conversion rate, where conversions are rare but trusted post-click activities, like purchases from the advertisers.

[2]Traffic entries tagged by filters are logged in abusive log files. Both trusted and untrusted traffic entries exist in the log files. Only untrusted entries exist in the abusive log files.

Given the above challenges, our contributions can be better laid down into the following three efforts:

1. building statistical models for size estimation in an autonomous, passive and privacy-preserving way from aggregated log files (§ 3),
2. devising a distributed efficient algorithm that examines previous size estimates, and produces a predicted size for each possible IP for a period p before p begins to mitigate the deleterious effects of lag (§ 4), and
3. setting traffic filtering thresholds based on the sizes without any manual intervention (§ 5).

The cycle and the interdependencies between these three processes is summarized in § 2.

2. THE SIZE ESTIMATION CYCLE

The cycle of size estimation and filtering is laid out in this section. The basic cycle consists of four processes that communicate via log files and size lookup tables. For period p, the inputs and outputs of the real-time traffic event logging, estimation, predictions, and real-time abuse detection processes are formalized in relations 1, 2, 3, and 4, respectively.

$$\textit{traffic}_p \xrightarrow{\textit{RT-Log(p)}} \textit{log-files}_p \tag{1}$$

$$\textit{log-files}_p \bowtie_{entry} \textit{abusive-log-files}_p \xrightarrow{\textit{Est(p)}} \textit{estimates-table}_p \tag{2}$$

$$\forall_{i=p-w-1}^{p-2} \textit{estimates-table}_i \xrightarrow{\textit{Prd(p)}} \textit{predictions-table}_p \tag{3}$$

$$\textit{traffic}_p \bowtie_{IP} \textit{predictions-table}_p \xrightarrow{\textit{RT-Abuse-Dtct(p)}} \textit{abusive-log-files}_p \tag{4}$$

Real-time logging, denoted *RT-Log(p)* in rel. 1, finalizes the traffic $\textit{log-files}_p$ as $p+1$ starts. Next, the $\textit{log-files}_p$ are consumed, among other input, by the estimation process, *Est(p)*, to produce the $\textit{estimates-table}_p$ mapping IPs that issued traffic during p to their estimated sizes (rel. 2). Next, the algorithm for predicting sizes, *Prd(p + 2)*, consumes the *estimates-tables* from a sliding window of length w periods[3], $p-w+1$ through p, to produce the $\textit{predictions-table}_{p+2}$. We assume that before the beginning of $p+2$, this prediction process, *Prd(p+2)*, completes and produces the $\textit{predictions-table}_{p+2}$, mapping IPs to their predicted sizes of period $p+2$.

The *estimates-tables* that contributed to $\textit{predictions-table}_p$ are shown in rel. 3. The $\textit{predictions-table}_p$ is used by the real-time abuse detection process, denoted *RT-Abuse-Dtct(p)* in rel. 4, to produce the $\textit{abusive-log-files}_p$ for p. The $\textit{abusive-log-files}_p$ contain the IDs of the traffic entries in $\textit{log-files}_p$ identified as abusive. The $\textit{abusive-log-files}_p$ are joined with the $\textit{log-files}_p$ by *Est(p)* to disregard the abusive traffic entries, and produce estimates based solely on legitimate traffic (rel. 2). While this joining makes estimation exclusively based on non-abusive traffic, care should be taken to avoid over-filtering of legitimate traffic.

This over-filtering caveat is best clarified by an example. Let IP 10.1.1.1 be stable at an estimated size of 1 for the periods $p-w$ through $p-1$, and then suddenly observes a flash

[3]The length, w, of the estimates window should be long enough to span cycles in the activities of the IPs such that *Prd(.)* considers legitimate cyclic size changes. Conversely, w should not be excessively large not to include very old sizes unrepresentative of future sizes. In our system, the estimates window was set to span several weekly cycles.

crowd during period p. *Prd(p+1)*, which runs during period p, is agnostic to this flash crowd and predicts a size of 1 for period $p+1$. Hence, *RT-Abuse-Dtct(p+1)* filters the majority of the traffic from 10.1.1.1. When the *log-files*$_{p+1}$ and the *abusive-log-files*$_{p+1}$ are joined, most of the traffic from this IP is not considered for estimation, and *Est(p + 1)* underestimates its size. Since the *estimates-table*$_{p+1}$ are fed back into *Prd(p+3)*, 10.1.1.1 continues to have a small predicted size, and to be overfiltered in $p+3$. To mitigate over-filtering caused by this hysteresis loop, only the egregiously abusive traffic is disregarded for the purpose of estimation[4].

The estimation and prediction phases have been assumed so far to run together in less than $l = |p|$, the period length. This introduced a *lookahead delay* of $2l$. That is, the output of *RT-Log(p)* is not used by *RT-Abuse-Dtct(.)* before $p + 2$. The longer the *lookahead delay*, the higher the chance of filtering based on inaccurate sizes. Enhancements to the basic cycle to reduce the *lookahead delay* are discussed in Appendix A. We next discuss the *Est(.)*, *Prd(.)* and *RT-Abuse-Dtct(.)* phases in § 3, § 4 and § 5, respectively.

3. ESTIMATION DETAILS

In the sequel, a user is defined as an entity that generates average activity of a trusted human for a specific application over a particular time period of length l. It is assumed that cookie IDs in the log files temporarily identify trusted users. Models of the average activity built from cookie IDs are influenced by the noise of users sharing or frequently clearing cookies. The log entries of one cookie ID may show the activity of one or more users, or part of the activity of one user. Thus, these cookie IDs do not perfectly represent real users. This phenomenon is a part of the trusted user activity, and is hard to separate from the log files.

Moreover, there is natural variance in the users' activity that can result in size estimation errors. However, this risk is lower for large IPs, where the absolute estimation errors are operationally more significant. The estimation error decreases as the IP traffic increases, as verified in § 3.4.

Next, two main classes for building models are presented: the rate estimators (§ 3.1) and the features diversity estimators (§ 3.2). Combining estimates is described in § 3.3.

3.1 The Rate Estimators Class

This class of estimators relies on the activity rate of an IP to estimate the number of users behind it. For any activity, if the trusted-user activity rate is verified to follow a Poisson distribution, then from the properties of the distribution, the size of an IP can be estimated based on its activity rate.

For a Poisson distribution with rate λ, the probability of having k log entries from a single user within one period is given by $f(k, \lambda l) = \frac{(\lambda l)^k e^{-\lambda l}}{k!}$. The sum of Poisson random variables with parameters $\lambda_1, \ldots, \lambda_M$ is a Poisson random variable with $\lambda = \sum_{i=1}^{M} \lambda_i$. For the estimated activity, the average trusted-user rate, λ_m, can be used to estimate the sizes of IPs. An IP with M users is expected to have a rate of $M \times \lambda_m$. More formally stated, the Maximum Likelihood Estimator (MLE) of the number of users behind an IP is $\frac{\lambda_{IP}}{\lambda_m}$, where λ_{IP} is the rate of activity of the IP.

To estimate the average unknown trusted-user rate of activity, the straightforward way is to calculate the MLE of λ_m, the average number of activity entries per trusted cookie per time period. A more practical method that is less susceptible to outliers is to construct the distribution of rates, and use its median or mode as λ_m. An empirical c-confidence interval on estimated size is given by $[\frac{\lambda_{IP}}{\lambda_{hi}}, \frac{\lambda_{IP}}{\lambda_{lo}}]$, where c is the fraction of the trusted users' rates between λ_{lo} and λ_{hi}.

Taking the number of users who query Google as an example, Fig. 2 shows the query rate distribution of $\approx$ 100M highly trusted cookies[5]. A median of 7, and an interquartile interval $[\lambda_{0.25}, \lambda_{0.75}]$ of $[3, 24]$ can be used for estimation.

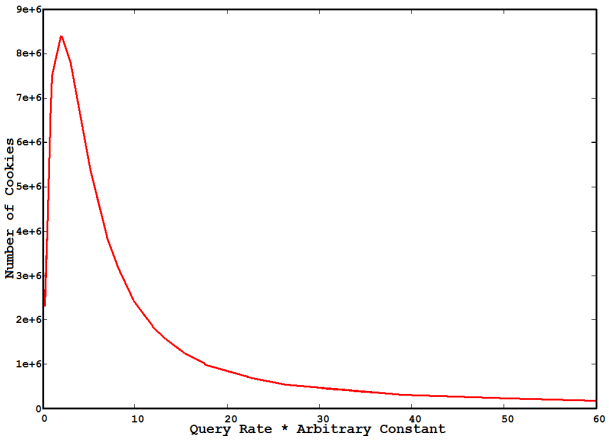


Figure 2: The query rate distribution PDF.

3.2 The Features Diversity Estimators Class

This class of estimators builds a regression model of the numbers of trusted cookies behind IPs, i.e., the baseline of *measured sizes*, as related to the *explanatory diversity* of their traffic feature(s). For example, after a linear model is built of how the *measured sizes* of IPs relates to their distinct queries, the same model can then be used to estimate the query size of any IP from its number of distinct queries.

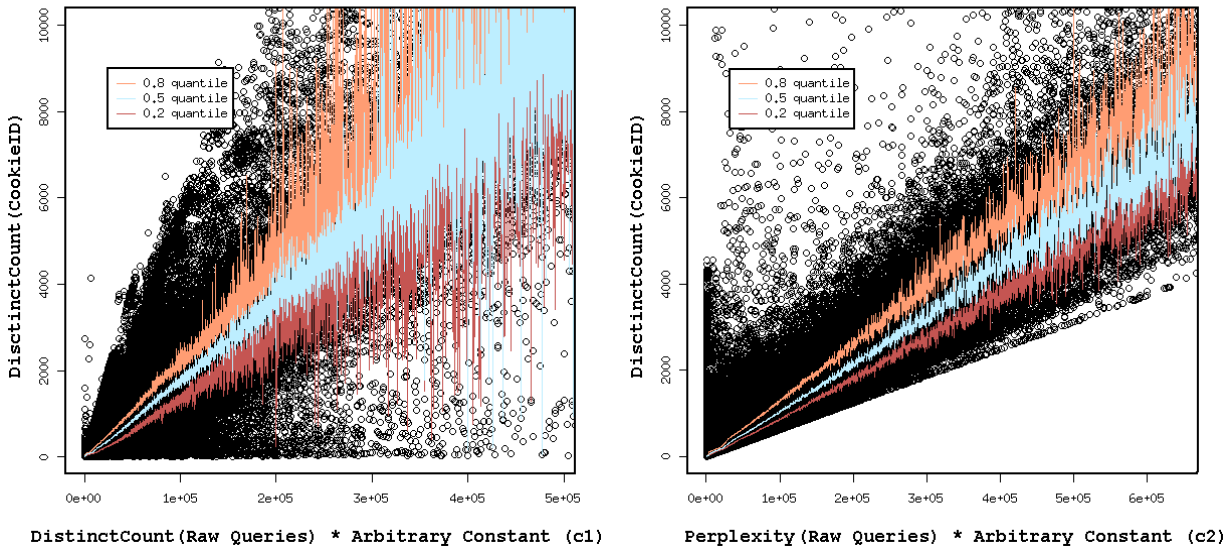
The *explanatory diversity* of feature(s) can be quantified in several ways. One simple way is counting its distinct values in the IP traffic. More sophisticated ways include calculating the perplexity and the compressibility of the sequence of the feature in the IP traffic. A feature, X (e.g., the query) in the traffic of an IP typically assumes several values, $x_1, x_2, \ldots$ (all the possible query phrases). The perplexity of a feature is calculated as $Perp(X, b) = b^{H_b(p)}$, where b is some base and $H_b(p) = \sum_x p(x) \log_b(\frac{1}{p(x)})$ is the entropy of the distribution of feature X in the IP traffic[6].

A training sample of $\approx$ 10M IPs, was collected to build a linear regression model of the number of users as related

[4]We have defined egregious traffic as the traffic that was filtered by another filter already deployed at Google. However, as a guideline if this is the only deployed filter, egregious traffic can be defined as the traffic filtered using a threshold h times higher than the normal threshold for the size of the source IP, where $h > 1$. Selecting h involves a trade-off. As h increases, filtering abusive traffic is reduced, which could later contribute to overestimating sizes of abusive IPs. Building attacks slowly over time exploits this vulnerability. As h decreases, the filter becomes less vulnerable, but produces more false positives since the estimation cycle becomes less responsive to unforeseen legitimate changes in sizes.

[5]Due to the sensitive nature of the exact distribution, the rate is scaled by an arbitrary constant.

[6]Perplexity was verified on several datasets to exhibit linear relationship with the numbers of trusted cookie IDs behind IPs (*measured sizes*). Entropy does not exhibit this quality.



(a) The distinct count of queries scattered with user count.

(b) The perplexity of queries scattered with user count.

Figure 3: The scatter plots of query diversity and the number of users.

to the query diversity quantified using distinct counting and perplexity. The data used to build the regression model for query sizes is plotted in Fig. 3. In Fig. 3, each circle represents one, or multiple overlapping sampled IP(s). Each circle shows the distinct number of trusted cookies querying Google, and the distinct count (Fig. 3(a)) and perplexity (Fig. 3(b)) of the queries issued by these trusted cookies[7].

Because models are built from log files aggregated at the level of IPs, and the overwhelming majority of IPs have very few trusted cookies behind them, sampling noise can cause issues. If a random training sample is selected, the few IPs with large *measured sizes*, i.e, the IPs with numerous trusted cookies, can be easily missed out. To avoid under-representing IPs with large *measured sizes*, stratified sampling is used [16]. IPs are bucketed into disjoint classes by their *measured sizes*. The number of samples from each class should represent this class in the global sample by the same proportion of that class in the global IPs population.

From Figures 3(a) and 3(b), some observations can be made about the stratified sample of IPs. First, the relationship between the *measured sizes*, i.e, the number of trusted cookies, and query diversity shows high heteroscedasticity. That is, the variance of the *measured sizes* increases with the distinct count (Fig. 3(a)) and perplexity (Fig. 3(b)) of queries. To build the model, instead of ordinary least squares linear regression, quantile regression [8] is used[8], since it is less susceptible to outliers.

The second observation is the data density decreases almost exponentially as the *explanatory diversity* increases. Using quantile regression to build the regression model on all the data yields models whose accuracy highly favors the dense areas of the low values of the *explanatory diversity*, and demonstrates low accuracy bias towards IPs with large *measured size*, where the accuracy is more critical. To build an estimation model that does not strongly favor dense regions, the high disparity in the data density should be normalized by giving all values of the *explanatory diversity* roughly the same weight. For a quantile, q, let the *q-quantile-curve* be the set of the q-quantile points of the *measured sizes* across all values of the *explanatory diversity* on the x-axis. Let the slopes of the lines that best fit the *q_{lo}-quantile-curve*, 0.5-*quantile-curve* and *q_{hi}-quantile-curve* be $\theta_{q_{lo}}$, $\theta_{0.5}$ and $\theta_{q_{hi}}$, respectively, and $q_{hi} - q_{lo} = c$. Then, the estimated size of an IP whose *explanatory diversity* value is ρ_{IP} is $\rho_{IP} * \theta_{0.5}$; the size of the IP is in the range $[\rho_{IP} * \theta_{q_{lo}}, \rho_{IP} * \theta_{q_{hi}}]$ with confidence c.

For example, in Fig. 3, the *quantile-curves* of 0.2, 0.5 and 0.8, are plotted. Each point on the 0.5-*quantile-curve* in Fig. 3(a) constitutes the median of the *measured sizes* on the y-axis for its corresponding value of distinct query count on the x-axis. The number of unique queries from an IP scaled by $\theta_{0.5}$ is its estimated query size.

3.3 Combining Estimates

For each IP, individual estimates are calculated based on its traffic rate and diversity, as discussed in § 3.1 and § 3.2. Even more, these two classes of estimators produce multiple individual estimates if the application has multiple "sub-activities". Taking the Google social network service as an example, a user can do several sub-activities like updating status, or sharing links. Each one of these sub-activities can be used for size estimation based on its rate and diversity.

When building the estimation models, the individual estimates are linear-regressed against the baseline of *measured sizes* to calculate the *combining weights* that are later used to calculate the *overall estimated sizes* when doing estimation. The *overall estimated sizes* are expected to have higher confidence and less error variance than any individual size estimate; and should serve as a general purpose size for the application whose activity is subject to estimation.

[7]Due to the sensitive nature of the exact distribution, the x-axes of Fig. 3 are scaled, and the perplexity is calculated with two bases, $b1 \neq b2$, as $Perp(X, b_1, b_2) = b_1^{H_{b_2}(p)}$.

[8]A q-quantile regression applied to a dataset fits a hyperplane that lies under roughly q of the points.

Preliminary filtering can be carried out at this stage. Since each IP typically has multiple individual estimates, high variance in these estimates is a strong signal for focusing on a specific activity, which is usually abusive. For instance, IPs that have noticeably high rates of issuing friend requests, and uncommonly low rates of status updates could be suspicious. IPs with high variance in their individual estimates are ignored when building the estimation models and have their traffic tagged as abusive during the filtering phase. The simplest method for setting a threshold on the variance is to measure the variances of all the IPs, calculate the standard deviation of the variances, and set the threshold at 2 or 3 times the standard deviation. More robust methods for setting this threshold are currently under investigation.

3.4 Gauging Estimation Accuracy

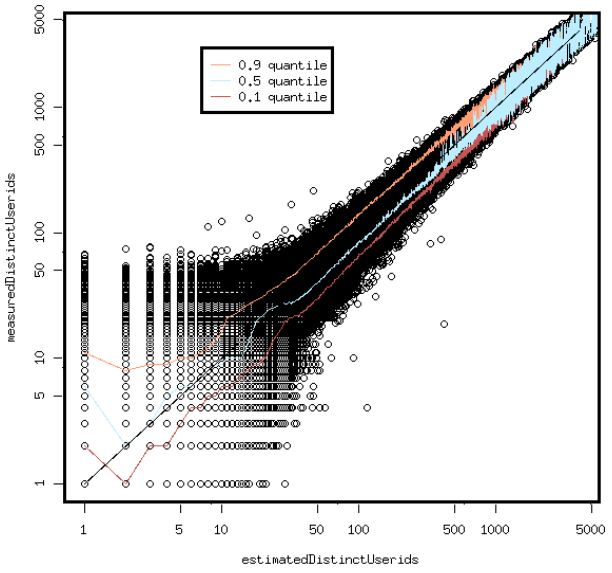


Figure 4: The estimated vs. measured query sizes.

Assessing the accuracy of the estimation process is done by using the estimation models on a testing set that is different from the training set. Only the traffic from the trusted cookies are used to produce the *overall estimated sizes* of the IPs in the testing set. These estimates are compared against the baseline *measured sizes*, the number of trusted cookies behind these IPs. For the purpose of modeling and gauging accuracy, only the traffic from the trusted cookies are used to produce the size estimates, and traffic from the non-trusted cookies is ignored. However, in reality, all the traffic is used to estimate the total number of users, and not only the trusted cookies users.

The sizes and the 0.1, 0.5 and 0.9 *quantile-curves* are plotted in Fig. 4 with logarithmic axes, where each circle represents one, or multiple overlapping IP(s). The line passing though the point $(1, 1)$ with slope 1 (drawn in black) represents perfect estimation. The median *quantile-curve* is almost overlapping with perfect estimation for estimates above 1, and 80% of the estimates are close to perfect estimation.

While from a statistical perspective it is undesirable to have estimation accuracy biases, it was purposeful for larger IPs, where the relative estimation errors are operationally more significant, to be more accurate. To achieve that, the sampling of the training IPs was stratified, and quantile regression on the *quantile-curves* was used to fit the models.

4. PREDICTION DETAILS

As discussed in § 2, the *estimates-tables* produced by the *Est(.)* phase over a window of size w are input to the *Prd(.)* phase. To reduce the *lookahead delay*, the prediction algorithm was developed with high focus on efficiency.

4.1 The Size Prediction Alternative Approaches

Predicting the size of every possible IP based on its previous size estimates is a gigantic web-scale time series analysis problem.

Typically, a time series prediction is carried out by doing a weighted average of the previous w values, where the weights are usually calculated offline using regression analysis. This is effective with time series with high autocorrelation where the current state is a function of previous states and white noise. This is customarily combined with smoothing, such as moving averages, to reduce noise, yielding the commonly known autoregressive moving average (ARMA) models [7].

Other alternatives for predicting sizes include nonparametric regression, like Spline curves [6]. The accuracy of these techniques depends largely on the number of iterations used to fit some model. This computational cost is still acceptable if the modeling is done offline.

There are several challenges with applying conventional time series procedures for predicting sizes of IPs. First, the time series of sizes are non-stationary. The time series of each IP does not follow the same distribution over time due to, among other factors, the reassignments of over 30% of the dynamic IPs every one to three days [19]. Such abrupt and unforeseen changes to sizes cause lack of stationarity, which limits the application of some techniques, like ARMA.

The second and bigger challenge is the estimates of each IP form a time series that should be analyzed to produce a prediction for this IP. Given the high heterogeneity in the behavior of IPs according to numerous factors including their assignments, time-zones and sizes, building a one-size-fits-all predictive model based on a sample of IPs becomes impracticable. Therefore, a specialized and efficient prediction algorithm is sought. While the time series literature does not strictly apply to this problem, the *PredictSizes* algorithm, described in § 4.4, employs some concepts from seasonal autoregressive integrated moving average (ARIMA) models.

At a high level, for each IP, *PredictSizes* predicts its size in isolation based on its latest w size estimates. For each IP, the *PredictSizes* algorithm performs three main functions. First, it analyzes the periodicity of the size estimates, since it has been consistently observed that the activity of IPs is periodic (§ 4.2). Second, for each periodicity, *PredictSizes* analyzes a sliding window of estimates and seeks their representative stable size by doing iterative variance reduction until the estimates lie within an acceptable confidence interval (§ 4.3). Third, it combines the estimates of all periodicities.

4.2 Considering Multiple Size Periodicities

Considering the periodicity of IP activity is imperative. Periodicities of the sizes of the IPs were discovered by selecting a sample of IPs, and applying Discrete Fourier Transform to each. The terms with the highest coefficients are the periodicities used by the *PredictSizes* algorithm [3]. For the vast majority of IPs, most of their sizes were noticed to have diurnal and weekly periodicities. These periodicities were especially clear for the IPs of school districts and large institutes, as informed by the Whois databases [18].

PredictSizes fetches the estimates of several periodicities, e.g., diurnal and weekly, for each IP to produce its prediction. For n periodicities, $s_1 < s_2 < \cdots < s_n$, *PredictSizes* considers the most recent w_i estimates s_i periods apart, for $1 \leq i \leq n$. For example, to estimate the sizes of IPs in six hours with all the sliding windows having length 10, $s_1 = 1$, $s_2 = 4$ and $s_3 = 28$, *PredictSizes* is considering the last 10 six-hour contiguous estimates, as well as the same-slot estimates of the last ten days and the last ten weeks.

4.3 Iterative Variance Reduction

PredictSizes deals with the sizes time series of each periodicity of each IP in isolation. It then combines the predictions from all the periodicities of an IP as discussed in § 4.4.

For time series predictions, it is typical to do simple trend analysis using simple linear regression with time as the explanatory dimension to show consistent increase or decrease over time (allowing for some white noise) [7]. The trend is then used for extrapolation. However, based on analysis of numerous IPs, time series of size periodicities almost never show strong trends within the window of estimates used for predictions. Moreover, using trend analysis hurts IPs that have drastic size change, since false trends result in erroneous predictions. Hence, *PredictSizes* assumes a stable value for each periodicity time series. The stable value, the representative statistic on the time series, is calculated using the *StableSize* function and is produced as the prediction.

For simplicity, the *StableSize* algorithm deals with each periodicity time series as a set. For each time series, *StableSize* does iterative variance reduction by removing outlier estimates that contribute the most to the variance until the ratio of the width of the confidence interval to the mean falls below a specific bound. The truncated mean of the remaining sizes is declared the stable size of this time series.

At each iteration, *StableSize* calculates the standard deviation, mean and the width of the c-confidence interval on the mean of the time series. The element that contributes the most to the variance is the farthest from the mean. This element can be identified in constant time by checking which of the maximum and the minimum elements are farther from the set mean and deleting it in each iteration. Each time an extreme element is deleted, the new mean and variance are incrementally accurately updated in constant time [17]. The most costly process is then identifying the extreme elements, which can be done efficiently using a minmax heap.

The algorithm fails if the time series exhibits little stability, due to abrupt size changes or due to the weakness of this periodicity compared to others. It bails out once the fraction of the discarded elements exceeds some threshold.

4.4 The PredictSizes Algorithm

Since *PredictSizes* deals with each periodicity of each IP separately, it can be massively parallelized using the Mapreduce framework [14], as the size estimates are stored in files sharded by the period IDs and IPs. The algorithm combines all the stable sizes of all the periodicities using a *Combiner* function that also does sanity checks on the predicted sizes.

The main factor that influences the choice of the *Combiner* is the loss function of the predictions, which is application-dependent. In its simplest form, a *Combiner* can be a simple statistic, such as the mean, truncated mean, median, max or min. For instance, when sizes are used for service optimization, the mean statistic minimizes the expected loss under the mean squared error loss function. Another alternative for *Combiner* functions is using a weighted average of the stable sizes, where the weights are inversely proportional to the fraction of size outliers (extreme estimates) discarded by the *StableSize* function for each periodicity. More involved analysis entails doing a regression of the size estimates of a particular period (as the expected *predicted size*) as related to the stable sizes from individual periodicities (as *explanatory sizes(s)*). However, such a regression-based *Combiner* can be easily influenced by the heterogeneity of IPs discussed in § 4.1, such as time-zones.

The *Combiner* algorithm ensures that the predicted size agrees with the stable sizes of all the periodicities. A simple solution was implemented that does two sanity checks. First, it checks that the predicted size is within some factor of the stable size for each periodicity. Second, it checks that the predicted size is within a specific quantile range of all the stable sizes. If the predicted size does not conform to the *Combiner* sanity check, the IP is deemed unstable, and no predicted size is produced for it. In our experiments, these simple sanity checks proved to be very effective in detecting abrupt legitimate size changes early on, and hence reducing the false positives caused by over-filtering legitimate traffic.

4.5 Evaluating Predictions

To evaluate the proposed predictions algorithm, an experiment was run on three months worth of query data log files. Two metrics were measured. The first metric is, for every period p, the agreement of the predicted sizes of the IPs with their estimated sizes during p (§ 4.5.1). The second metric is the *coverage*, the ratio of IPs in the traffic in period p that had predictions (§ 4.5.2).

4.5.1 Prediction Accuracy

To assess the prediction accuracy, a random sample of 10M IPs was collected. The relative ratio, *predicted size / estimated size*, is shown in Fig. 5, where each circle represents one, or multiple overlapping IP(s).

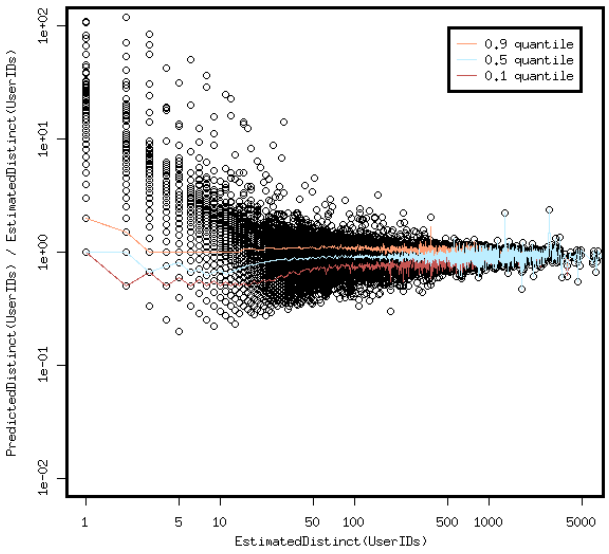


Figure 5: The relative ratio in predicting query sizes by the estimated sizes.

98% of the absolute errors are between −4 and 2, and 54% of the predictions are exact. The mean absolute error

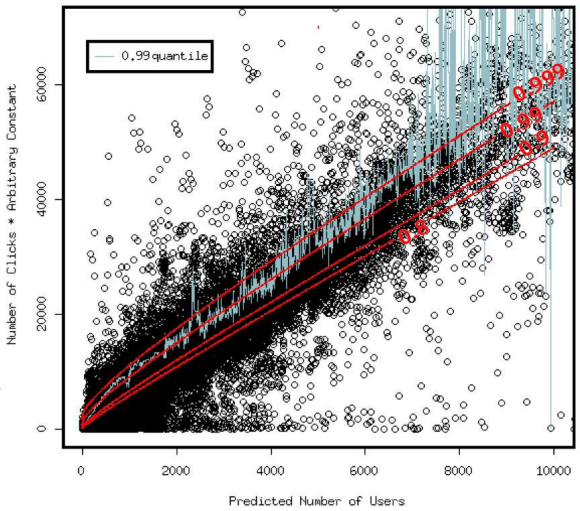


Figure 6: The CDF of a given number of clicks from a given number of users in red lines. The empirical CDF of a given number of clicks from a given estimated number of users in blue.

is -0.149. All the quantiles with a step of 0.001 were calculated. The topmost four 0.001 quantiles are 5282, 5, 4 and 3, and the bottommost quantiles are -6870, -12, -8, -7. Based on Whois databases, the IPs that caused the largest absolute errors belonged to large commercial ISPs with several netblocks and diverse customer bases. These IPs probably changed sizes due to reassignment.

From Fig. 5, among all the predicted sizes, 98% were within a factor of 2 of the estimated size. The topmost four 0.001 quantiles of the relative ratios were 5283, 4, 4 and 3, and the bottommost quantiles were 0.2, 0.4, 0.4 and 0.44.

Since the largest relative ratio was as high as 5283, it was concerning that reassignments of IPs and flash crowds can result in predicted sizes that are significantly smaller than the estimated sizes. The *IP-period* instances[9] showing this behavior in these three months were counted. No IP showed this behavior for more than 2 periods, owing to the 2 period *lookahead delay* discussed in the framework cycle (§ 2) and the *Combiner* algorithm sanity checks (§ 4.4). Only 33 *IP-period* instances, , composed of 21 IP's, had a relative ratio greater than 500. Some IP's spanned 2 periods. Only ≈ 4700 *IP-period* instances from only ≈ 3000 IPs had a relative ratio greater than 100. These anomalies are very insignificant compared to the number of IPs observed over three months.

The relative ratio is broken down by the estimated sizes in Fig. 5. The comparisons of measured vs estimated sizes are quantitatively similar to the comparisons of estimated vs predicted sizes. The line that passes through the y-axis point 1 with slope 0 represents perfect predictions. Clearly, the median *quantile-curve* is almost overlapping with the perfect predictions line for medium and large values of estimated sizes. In addition, 80% of the predictions lie in the vicinity of perfect predictions. Most importantly, the accuracy of the predictions increases as the estimated size increases, where the accuracy is more operationally desired.

4.5.2 Predictions Coverage

For predictions to be effective, they should have high *coverage*, i.e., a high ratio of the IPs in the traffic have predictions. There are several factors that contribute to the predictions coverage, such as the stability of the estimated sizes of the IPs, the diversity of the IPs that visit the application provider, the length of the estimation period and the length of the sliding windows of estimates used for prediction.

The coverage of the three-month experiments dropped below 95% on two days, and never dropped below 93%. The coverage of the click sizes was also examined. The click-coverage averaged around 65%, and never dropped below 61%. Since the number of queries is at least one order of magnitude more than the clicks, the query traffic is expected to come from more IPs, which better cover the IP space than the click traffic. Hence, the probability any IP generates traffic on two distinct *IP-period* is higher for queries than clicks, which explains the higher query-coverage.

5. THRESHOLD-BASED FILTERING

Thanks to the high accuracy and coverage of the estimation and prediction phases discussed in § 3 and § 4, respectively, a threshold-based traffic filter, *RT-Abuse-Dtct(.)*, was built based on the predicted sizes of the IPs. The ad click filter built at Google caps the number of clicks per IP at a specific threshold and filters any extra clicks.

5.1 Selecting the Filtering Threshold

The distribution of the number of trusted users (cookies) who generate N traffic log entries per period is reused when setting the filtering thresholds. Normalizing this distribution yields the PDF of the number of entries from the average trusted user. This PDF is convolved with itself M times to get the distribution of the number of entries from M users[10]. This distribution is then numerically integrated to produce a CDF of the number of entires from M users.

For the application of ad click filtering, the number of trusted users who click N times per period is used. The shape of this distribution is extremely similar to the corresponding query distribution in Fig. 2. The CDF contour lines for the 0.8, 0.9, 0.99 and 0.999 quantiles were calculated and are plotted in Fig. 6 in red for $1 \leq M \leq 10K$. The *q-quantile-curve* at size M indicates the number of clicks M trusted users should not exceed with probability q. For instance, from Fig. 6, 99% of the IPs with an estimated size of 2000 should generate clicks less than ≈ 1480[11]. Using a *q-quantile-curve* as the threshold, only $1-q$ of IPs with any given size have their clicks filtered. Therefore, q can be set based on how aggressive the filter is designed to be.

To check the combined accuracy of the estimation and prediction phases, the numbers of clicks from IPs with predicted sizes were analyzed. A sample of 1M IPs were selected at random, and their numbers of clicks are plotted in Fig. 6 with the 0.99-*quantile-curve* plotted in blue. Comparing the red and blue curves in Fig. 6 shows the theoretical and the empirical calculations of the 0.99-*quantile-curve* agree. This proves the accuracy of the proposed framework and the threshold selection mechanism.

Notice that while models exclusively built from the traffic of trusted users may be biased, reusing the distribution of the activity of the trusted users to select the filtering

[9]An *IP-period* instance is the size of one IP in one period.

[10]This convolution is done by performing Fast Fourier Transform (FFT) on the single-user distribution, raising the result to the M^{th} power, and taking the inverse FFT.

[11]The number of clicks is scaled by an arbitrary constant.

threshold mitigates such bias. Practically, both modeling and traffic filtering are done in the domain of trusted users.

5.2 Filter Evaluation

A sensitivity analysis on the parameter q was carried out. The value of q was varied and the filtered traffic as well as the false positives were measured. For the purpose of this paper, the false positives are calculated as the ratio of conversion rate from the tagged clicks to the conversion rate from all the ad traffic[12]. A conversion is a trusted post-click activity, like a purchase from the advertisers. While conversions are useful for evaluating filters, they are less useful for directly detecting fraud in real time, since they are scarce and delayed events.

When q was set to 0.999, the filtering rate was significantly higher than $1 - q$. Most of the filtered traffic entries were false positives. As q approached 0.99, the filtering rate was slightly higher than $1 - q$, with a low ratio of false positives. While the absolute number of false positives did not decrease as q decreased, the ratio of the false positives to the filtered traffic dropped drastically. As q approached 0.95, both the absolute number of false positives and their ratio to the filtered traffic increased quickly due to aggressive filtering. Hence, the threshold quantile was set at 0.99.

The discovered false positives were consistently due to unforeseen reassignments of IPs and flash crowds. For instance, the top 10 IPs causing false positives had predicted sizes significantly smaller than their estimated sizes. These false positives were reduced significantly by detecting abrupt legitimate size changes early on using the prediction sanity checks discussed in § 4.4.

The overlap between proposed filters and existing filters is a useful metric customarily measured at Google. Multiple filtering layers provide defense in depth against abuse. Among the first 50M clicks tagged by the filter, 99.55% of them overlapped with other filters. While the marginal gain of this filter is small, the filter is highly valuable as a very accurate safety net and as a verifier of other filters.

The false negatives, the clicks tagged only by other filters, decreased monotonically with q. Most of the false negatives occurred due to two reasons: botnet attacks with few clicks per IP, and attacks from IPs with no predicted sizes. To address the first reason, we are currently generalizing this filter to multi-dimensions. We are investigating thresholds on the number of clicks per IP across some dimension(s), such as the query term. This allows for setting tighter thresholds at a finer granularity. For the second reason of false negatives, we analyzed the coverage of the predicted click sizes. While the coverage of the predicted click sizes was around 65%, the number of clicks whose IPs had predicted sizes was roughly 78% due to the non-uniform distribution of the number of clicks from different IPs. To increase the recall of the filter, we are currently evaluating using a default high threshold for IPs with no predicted sizes.

The filter was deployed on all the ad traffic at Google, and had a significant recall of all the clicks tagged by all the filters. Analyzing and monitoring the filtered traffic of the deployed filters is customary at Google to retune the thresholds based on the natural changes in global traffic patterns. Among these 50M clicks, the rate of false positives was $\approx 1.4\%$, which surpasses many other deployed filters.

[12]A significant ratio of advertisers share conversion information with Google.

For the sake of comparison, we simulated a naïve filter that does not consider the sizes of the IPs. It tags clicks from IPs based on a fixed threshold. It was difficult to force the naïve filter to have exactly the same recall of the developed filter. However, to achieve comparable recall, the threshold of the naïve filter had to be set for aggressive tagging, which yielded a false positive rate of 37%.

6. RELATED WORK

To the best of our knowledge, this is the first work that mines Internet traffic to estimate sizes of IPs. This work complements the prior work on the dynamics of IPs and classifying them [2, 4, 5, 9]. Only some of the recent work on combating abusive traffic based on the source IPs [1, 10, 11, 13, 15, 19, 20] have considered the nature of IPs, e.g., the rate of reassignment, but not the sizes of IPs.

The work that is most related to estimating sizes of IPs deals with counting the hosts behind NAT devices [2, 4, 9, 12]. Bellovin presented a technique for counting the hosts behind a NAT using the `IPid` field in [2]. The technique relies on the host operating system sequentially incrementing the `IPid` field for each successive packet. Hence, the number of unique `IPid` streams from a specific IP can be used as an estimate of the number of hosts. This technique is limited by the resolution of the `IPid` field. Even more, the `IPid` field is not incremented sequentially in modern operating systems. Furthermore, in the measurements reported in [4], less than 5% of the sampled NATs sent multiple `IPid` streams.

The focus of [4] was identifying middle-boxes and classifying them as NAT devices and Proxies by learning the internal IPs of the hosts using active web content. However, this technique underestimates the sizes by the ratio of users not collaborating with the research effort. It also fails if a NAT box has a hierarchy of NAT devices behind it, where collisions, and hence under-estimation can happen.

In [9], Kohno *et al.* investigated using clock skews for identifying hosts behind NAT devices, where the clock skew is the ratio of the actual clock frequency to the nominal one. Typically, the clock skew of a host is stable within 1-2 $\mu s/s$. Given that the difference between the clock skews of different hosts goes up to 50 $\mu s/s$, hosts can be fingerprinted by their clock skews. The number of hosts behind a NAT device can then be passively estimated by counting distinct fingerprints from this NAT device. However, the accuracy of this technique is highly limited by the collisions between clock skew identities. For instance, [12] could only extract 4-6 bits of host identification using clock skews.

This work is different in scope, goal and methodology from these in [2, 4, 9]. The proposed techniques are not restricted to NAT devices to achieve high IP coverage. The proposed framework neither probes machines for pseudo-identification, nor does it deploy active content, which can be intrusive, and is hence worrisome for some users. The goal of this work is filtering abusive traffic according to the source IPs, while preserving the user privacy.

7. CONCLUSION

This paper is an example of how to mine traffic at an aggregate level in order to balance combating abusive traffic and preserving user privacy. It shares our experience:

1. building statistical models for estimating the number of application users behind IPs,

2. devising a sizes prediction algorithm, and

3. setting traffic thresholds based on the predicted sizes.

The framework has been deployed on Google live traffic for combating click fraud. While sharing this new framework could be inspiring, details of the signals and parameters used were concealed to preserve their effectiveness.

The proposed framework is general, and can be applied to a wide spectrum of applications to estimate a population of users who are not permanently identified. The framework only assumes the existence of a body of trusted users, whose activities are used to build models for size estimation. These models are known only to the application/service providers. When applied for abuse detection, the framework leverages such knowledge to give an advantage to the providers over the attackers in the arms race. The traffic filtering is also general, since most of the filtering is threshold-based, and is hence agnostic to the traffic nature.

Our future work focuses on applying the framework to more types of abuse, such as excessive service sign-ups, which have less traffic and lower coverage in the IP space. One approach being explored is using IP classification techniques [5, 19, 20] to improve the filtering coverage in the lack of high IP space coverage in the traffic. In addition, we are currently exploring other more sophisticated abuse detection and combating approaches that compare how the distribution of sizes of IPs per entity, e.g., the query term, to the aggregate background distribution of sizes of IPs.

Acknowledgment

We thank Adel El-Atawy, Adrian Isles, Amgad Zeitoun, Algis Rudys, Christos Faloutsos, John Hawkins, Kourosh Gharachorloo, Michael McNally, Mohamed Elfeky and the rest of the ad traffic quality team for the useful discussions and help with the deployment.

8. REFERENCES

[1] D. Anderson, C. Fleizach, S. Savage, and G. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *Proceedings of the 16th USENIX Security Symposium*, pages 135–148, 2007.

[2] S. Bellovin. A Technique for Counting NATted hosts. In *Proceedings of the 2nd ACM SIGCOMM IMW Workshop on Internet measurment*, pages 267–272, 2002.

[3] P. Bloomfield. *Fourier Analysis of Time Series: An Introduction*. Wiley-IEEE, 2004.

[4] M. Casado and M. Freedman. Peering Through the Shroud: The Effect of Edge Opacity on IP-Based Client Identification. In *Proceedings of the 4th ACM/USENIX NSDI Symposium on Networked Systems Design and Implementation*, pages 173–186, 2007.

[5] M. Freedman, M. Vutukuru, N. Feamster, and H. Balakrishnan. Geographic Locality of IP Prefixes. In *Proceedings of the 5th ACM SIGCOMM IMC Conference on Internet Measurement*, pages 13–13, 2005.

[6] J. Friedman. Multivariate Adaptive Regression Splines. *Annals of Statistics*, 19(1), 1991.

[7] J. Hamilton. *Time series analysis*. Princeton University Press, illustrated edition, 1994.

[8] R. Koenker and K. Hallock. Quantile Regression. *Journal of Economic Perspectives*, 15(4):143–156, 2001.

[9] T. Kohno, A. Broido, and K. Claffy. Remote Physical Device Fingerprinting. In *Proceedings of the 26th IEEE S&P Symposium on Security and Privacy*, pages 211–225, 2005. An extended version appeared in the IEEE Transactions on Dependable and Secure Computing, 2(2):93–108, 2005.

[10] A. Metwally, D. Agrawal, and A. El Abbadi. DETECTIVES: DETEcting Coalition hiT Inflation attacks in adVertising nEtworks Streams. In *Proceedings of the 16th WWW International World Wide Web Conference*, pages 241–250, 2007.

[11] A. Metwally, F. Emekçi, D. Agrawal, and A. El Abbadi. SLEUTH: Single-pubLisher attack dEtection Using correlaTion Hunting. *Proceedings of the VLDB Endowment*, 1(2):1217–1228, 2008.

[12] S. Murdoch. Hot or Not: Revealing Hidden Services by their Clock Skew. In *Proceedings of the 13th ACM CCS conference on Computer and Communications Security*, pages 27–36, 2006.

[13] T. Peng, C. Leckie, and K. Ramamohanarao. Proactively Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring. In *Proceedings of the 3rd International IFIP-TC6 Networking Conference*, pages 771–782, 2004.

[14] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the Data: Parallel Analysis with Sawzall. *Scientific Programming*, 13(4):277–298, 2005.

[15] A. Ramachandran, N. Feamster, and S. Vempala. Filtering Spam with Behavioral Blacklisting. In *Proceedings of the 14th ACM CCS Conference on Computer and Communications Security*, pages 342–351, 2007.

[16] G. Snedecor and W. Cochran. *Statistical Methods*. Wiley, John & Sons, Incorporated, eighth edition, 1991.

[17] D. West. Updating Mean and Variance Estimates: An Improved Method. *Communications of the ACM*, 22(9):532–535, 1979.

[18] Whois.net. Domain Research Tool. http://www.whois.net.

[19] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How Dynamic are IP Addresses? In *Proceedings of the 22nd ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 301–312, 2007.

[20] L. Zhuang, J. Dunagan, D. Simon, H. Wang, and J. Tygar. Characterizing Botnets from Email Spam Records. In *Proceedings of the 1st Usenix LEET Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, 2008.

APPENDIX

A. CYCLE ENHANCEMENTS

Some enhancements are applicable to the basic cycle in § 2 to reduce the *lookahead delay* introduced by the estimation phase. First, we introduce $Est_d(p)$ that divides the periods into d sub-periods, $p_1, \ldots, p_d$ of length $\frac{l}{d}$. Hence, preliminary processing can be done once a sub-period ends, and the intermediary results from all the d sub-periods can be combined together into $estimates\text{-}table_p$. The lower bound on the *lookahead delay* is hence reduced from $l\left(1+\left\lceil\frac{|Est(.)|+|Prd(.)|}{l}\right\rceil\right)$ to $l+(l/d)\left\lceil\frac{|Est_d(.)|+|Prd(.)|}{(l/d)}\right\rceil$[13].

The estimation period can also be defined as a sliding window over sub-periods. Hence, $d-1$ sub-periods would be shared by any two consecutive periods. This amortizes the preliminary processing on several cycles. As d increases, $Est_d(.)$ is expected to run in less time than $Est(.)$ as long as the benefits of producing preliminary results and amortizing their processing outweigh the cost of combining them.

The main advantage of the sliding window model is producing estimates roughly d times more frequently than the basic cycle. Let $PrelimEst_d(.)$ be the preliminary processing of any sub-period, and $Comb_d(.)$ be the combining of d the sub-periods results, then the sliding window model reduces the *lookahead delay* to $(l/d)\left(1+\left\lceil\frac{|PrelimEst_d(.)|+|Comb_d(.)|+|Prd(.)|}{(l/d)}\right\rceil\right)$. Therefore, for low traffic-volume applications, l can be increased proportionally to produce sound estimates from more traffic. Experimentation is needed to set the value of l and d based on the IPs' coverage in the traffic, and the processing power available.

[13]Each process run time is assumed consistent over time.