Peering Through the Shroud:

The Effect of Edge Opacity on IP-Based Client Identification

Martin Casado and Michael J. Freedman
Stanford University
http://illuminati.coralcdn.org/

Abstract

Online services often use IP addresses as client identifiers when enforcing access-control decisions. The academic community has typically eschewed this approach, however, due to the effect that NATs, proxies, and dynamic addressing have on a server's ability to identify individual clients.

Yet, it is unclear to what extent these edge technologies actually impact the utility of using IP addresses as client identifiers. This paper provides some insights into this phenomenon. We do so by mapping out the size and extent of NATs and proxies, as well as characterizing the behavior of dynamic addressing.

Using novel measurement techniques based on active web content, we present results gathered from 7 million clients over seven months. We find that most NATs are small, consisting of only a few hosts, while proxies are much more likely to serve many geographically-distributed clients. Further, we find that a server can generally detect if a client is connecting through a NAT or proxy, or from a prefix using rapid DHCP reallocation. From our measurement experiences, we have developed and implemented a methodology by which a server can make a more informed decision on whether to rely on IP addresses for client identification or to use more heavyweight forms of client authentication.

1 Introduction

Faced with an ever increasing amount of unwanted traffic, Internet services must be able to differentiate clients in order to enforce access-control decisions. One traditional approach is to use the client's IP address as an identifier. For instance, if a client is exhibiting malicious behavior—such as attempting to post spam messages to online blogs or trolling ssh servers for weak passwords—a server may add the client's IP address as a firewall rule that will prohibit further connections from the offending address.

Unfortunately, middleboxes (both layer-3 network address translation (NAT) devices and layer-5 proxies) and dynamic IP renumbering due to DHCP challenge the utility of using IP addresses as client identifiers. Given the previous example, adding a firewall rule for an IP address due to abusive behavior (blacklisting) can deny system access to many users that may share that IP, either simultaneously via a middlebox or over time via DHCP. Similarly, when IP addresses are naively used to allow access to a resource (whitelisting), an open proxy from an authenticated domain, such as a university, can enable Internet-wide access to the protected resource. We refer to this phenomenon of edge technologies obscuring a client's identity at the network level as *edge opacity*.

Due to these apparent operational issues, many websites have moved from using IP addresses as identifiers to requiring some form of registration, authentication, and then application-level identity checks. For example, it is not uncommon for wikis or web forums to require their users to register, login, and then present HTTP cookies on each access. Yet, these extra steps can be a usability hurdle and generally require more server-side resources.

Because of these concerns, other websites continue to use IP whitelisting and blacklisting as the basis for managing client access. And yet, as the extent of edge opacity remains unquantified, the impact of using IP addresses as client identifiers is uncertain. Thus, server operators are making uninformed decisions regarding the best way to use IP addresses as client identities.

The goal of our work is to help a server determine if the IP address of an incoming client is a useful identifier for access-control decisions. We do so by exploring the extent to which edge opacity obscures a server's view of its clients. Specifically, we consider the following questions:

1. To what extent does edge opacity from NATs, proxies, and DHCP prevent accurate IP-based client identification?
2. Is it possible for a server to determine when IP-based filtering is insufficient and stronger methods of identification are required?

To answer the first question, we performed a large-scale study measuring 7 million unique web clients from 177 million HTTP requests over a seven-month period. We found that while the majority of clients are behind NATs (roughly 60%), NATs are small and localized—with most serving fewer than 7 hosts. Furthermore, clients using DHCP generally rotate their IP addresses on the order of several days, limiting the impact of dynamic addressing on short-lived access-control decisions. Web proxies,[1] on the other hand, while fewer in number (roughly 15%), often serve large and commonly geographically-diverse client populations.

To answer the second question, we introduce, analyze, and implement a set of techniques with which a web server can detect with high accuracy if a client is using a proxy or NAT, or whether it belongs to a prefix with rapid IP reallocation. Because of their potential to host large client populations, we mainly focus on proxy detection, proposing additional methods to determine a client's distance from its proxy and to disambiguate multiple clients behind a single proxy.

The main contributions of this paper are as follows.

- We introduce and demonstrate the utility of using active content as a measurement vehicle for understanding the Internet's edge (§3).
- We present the results of a large-scale measurement study which helps to quantify the extent and characteristics of NATs, proxies, and DHCP usage (§4). We believe these results are original and have independent academic interest.
- From our measurement experience, we derive and implement a methodology to aid Internet services in determining whether or not a client's IP address is a useful identifier (§5).

2 Background and Limitations

2.1 IP vs. application-level identification

There are a number of practical benefits to using IP addresses as the basis for enforcing access controls in today's Internet. IP-based filtering, ACLs, and rate-limits are all standard on firewalls and routers. Further, these IP-based enforcement mechanisms typically operate at carrier-grade line speeds. Filtering on IP addresses also allows an online service to "drop early," rather than waste CPU, memory, or bandwidth resources on traffic that will be eventually dropped via heavier-weight mechanisms.

Application-level identification methods (greylisting), on the other hand, can be used to differentiate clients when IP addresses are not sufficient. However, these methods often come at a cost. Many greylisting methods put a burden on users, for example, requiring them to answer CAPTCHAs [32] or to go through a separate sign-on step. Application-level identification is also more resource intensive for the server. Whereas IP blacklisting can drop packets before they ever reach the server, registration requires that the server handles a client's session prior to its decision whether to accept the client.

We note that if the server's greylisting scheme is effective, an attacker has no incentive to masquerade as a proxy. Most obviously, greylisting may require the attacker to expend resources—*e.g.*, requiring some cryptographic computations or additional time for "tarpitted" TCP connections to complete—better spent attacking other systems. Furthermore, if the time required to identify malicious behavior and blacklist the client is greater than that needed to greylist a proxy, an attacker has a disincentive to appear as a proxy: immediate greylisting would provide the attacker with fewer opportunities to attack the system. This argument does not necessarily hold if the attacker is attempting to launch a denial-of-service attack on the server; we consider this case in §2.3.

2.2 Other uses for edge opacity detection

With the exception of this subsection, this paper only discusses using IP addresses to enforcing access controls. In practice, IP addresses are used by servers in a variety of other settings, including geolocation and fraud detection, which we review here.

Websites use geolocation in both content personalization and access control contexts. For example, Major League Baseball uses Quova's GeoPoint server [20] to ensure that games are not webcast to subscribers subject to blackout restrictions. Yet end-users can often circumvent such restrictions by using proxies (indeed, we have personal experience with clients using CoralCDN for precisely this reason [3]); accurate proxy detection is needed to prevent such behavior. Furthermore, if a server can peer through a client's proxy to determine its true location (§5.1), advertisements can be more accurately targeted.

An important aspect of combating online fraud is to detect suspicious IP-based usage: *e.g.*, when a California-based bank customer accesses his online account from Romania, or when huge numbers of clicks for pay-per-click advertisements originate from a small IP range. Proxy detection can improve servers' decisions; for instance, by discovering when a fraudster is trying to proxy its banking session behind a U.S.-based IP address, or by recognizing the potential legitimacy of high rates of traffic originating from AOL's proxies. While efforts to correlate anomalous IP characteristics to fraud exist [10], their ability to detect and peer behind proxies is limited.

Our paper does not further address these two additional uses of IP-based identification. Indeed, evaluating the accuracy of a geolocation package or the security of a fraud-detection system is challenging and beyond the scope

[1] In this paper, we refer to address translation middleboxes that do not perform TCP termination as NATs. Layer-5 proxies, on the other hand, establish application sessions with both their clients and remote servers and therefore terminate their clients' TCP sessions.

Country	Unique hosts
United States	1,290,928
Japan	1,024,088
China	912,543
Germany	385,466
Great Britain	218,125
France	215,506
Canada	204,904

Table 1: Top 7 countries by unique hosts measured

of this paper. We describe these examples, however, to demonstrate why—if one were to deploy a geolocation or fraud-detection system (and many companies do exactly that [10, 12, 20, 26])—detecting proxies and other IP exceptions is important.

2.3 Limitations

In what follows, we present results from our efforts to characterize NAT, proxy, and DHCP usage and their effects on client identification. From March through September, 2006, our measurement servers handled over 177 million HTTP requests, originating from nearly 7 million unique hosts from 214 countries. Table 1 gives a sense of the dataset's international coverage, including large numbers of requests from the United States, Japan, China, and Europe. As we mention in 3.2, we collected measurements from web clients as they visited a wide variety of different sites serving largely different client demographics.

While these measurements are over a large and diverse population, we do not claim to provide a representative sampling of *all* Internet traffic.[2] Rather, our goal is to understand edge opacity from the viewpoint of a server operator: What is the extent to which its clients are behind middleboxes or experience rapid IP reallocation? Secondly, given this large body of measurement data, what techniques and heuristics can we develop to identify when a server operator should not use IP address and client configuration information for access control decisions?

We do not envision opacity detection to be useful to prevent denial-of-service attacks. An attacker with sufficient resources can resort to flooding with spoofed IP addresses, making any IP-based logic meaningless. Further, an attacker attempting an application-level DDoS could masquerade as a proxy and potentially afford herself more access to resources. However, we found that IP blacklisting will affect far fewer clients than we had originally anticipated (§4).

[2]For example, users of Internet cafes are probably more likely to use online email, chat, or game services, none of which we were able to measure.

3 Measurement Overview

3.1 Using active content for measurement

Our approach to studying the impact of NATs, proxies, and DHCP-based IP allocation relies on a variety of network- and application-level client characteristics. These include such properties as public and local IP addresses, round-trip-time measurements, system configuration parameters, and so forth. Unfortunately, much of this information cannot be captured through traditional measurement means alone, *i.e.*, active probing or passive traffic sniffing [7].

Instead, we leverage active web content as a means for gathering edge-network configuration state. We redirect oblivious web clients to a custom-built web server that serves web pages with active content (currently, javascript and/or a Java applet). The server then performs some limited passive analysis and cooperates with the executing code to measure and collect client configuration state, which is analyzed and added to our measurement database.

In contrast to active measurement, active content is executed at the application layer, generally *behind* NATs or proxies. Active content is in wide use by popular websites today; for example, users of Google's GMail application execute tens of thousands of lines of javascript. Thus, it is both almost universally supported and, at the same time, much less likely to generate abuse complaints normally associated with unsolicited scans of end-hosts [18]. (We have yet to receive a single abuse complaint.)

Furthermore, in contrast to expecting clients to download proprietary measurement code [13, 19], we target content providers. Web clients access our measurement platform either when requesting a web-beacon placed on popular websites (as of September 2006, 32 third-party sites have incorporated this web-beacon) or after being redirected from CoralCDN [9, 17]. While browser-based active content can collect strictly less information than client-executed proprietary code—insofar as it must operate within the confines of browsers' security policies—it is easier to gather large-scale measurements. For instance, the Dimes project [13] has collected measurements from roughly 11,000 agents from 5,000 users over 2 years of operation, almost three orders of magnitude less than our coverage.

3.2 Data collection methods

To cast a wide net, we collect information from two main sources. First, we built an online measurement community of content providers, allowing websites to track and compare their individual contributions. A wide variety of websites contributed to our measurements, including popular blogs, news sites, personal web-pages, education sites, hosting services, hobbyist sites, and public forums.

Second, we have instrumented CoralCDN [9, 17], a popular web content distribution network we run, to redirect a small percentage of its approximately 25 million daily requests through our measurement servers. Traffic from CoralCDN currently accounts for approximately two-thirds of our clients and its world-wide distribution.

Operators of servers can contribute to our measurement project in two ways. First, websites insert a transparent, embedded object into their web pages (the 1x1 pixel `iframe` "web-beacon" in Figure 1). When clients download these sites' pages, they subsequently request this web object from our measurement servers. Alternatively, a server integrates client-side measurements by redirecting link click-through traffic to our measurement servers, which causes clients to load our web-beacon before they are redirected back to the link's real destination. We used this latter approach for measuring clients of CoralCDN.

Whether through the web-beacon or a redirect, the client executes javascript that collects various configuration parameters, *e.g.*, browser version information, screen parameters, system languages, timezone settings, etc. Java-enabled clients may also execute a Java applet that creates a socket connection back to our measurement servers, from which the client grabs its local IP address and ephemeral port. Differences between the client's local IP address and its public IP address (as seen by our server) indicates the existence of an on-path middlebox.[3]

Of course, both the javascript and Java applet code are constrained by browser security policies, such as same-origin restrictions. Additionally, cognizant of privacy concerns, we have avoided collecting particularly invasive information (such as extracting browser cache history through CSS tricks).

Rather than explain the details of all our data-collection techniques upfront, we will describe these techniques incrementally as we consider various questions throughout the remainder of this paper.

3.3 Dataset coverage

This paper's dataset, described in Table 2, covers client data collected between March 3, 2006 through September 27, 2006. It includes nearly 7 million unique hosts, where a unique host is identified by a three-tuple of the client's public IP address, its local IP address if available, and its SYN fingerprint [27]. A smaller fraction of clients (1.1 million in all) executed our Java applet, due both to some third-party websites choosing a no-Java "diet" version of our web-beacon and the fact that only approximately 30% of our measured clients will actually execute Java.

[3]We can also use differences in the ephemeral ports to help detect transparent proxies and provide some insight into how the middleboxes are doing port re-mapping, although we do not include such analysis in this paper.

Unique targets	
Hosts measured	6,957,282
Public IPs	6,419,071
Hosts running Java	1,126,168
Hosts behind middleboxes	*73.8%*
Coverage	
IP Prefixes (per RouteViews)	85,048
AS Numbers (per RouteViews)	14,567
Locations (per Quova)	15,490
Countries (per Quova)	214

Table 2: Dataset statistics

In addition, we analyzed over 4 million requests from 540,535 clients that ran a heavier-weight version of the javascript code (collected between August 29 and September 27, 2006). This dataset, used in §4.3 and §5.1, included client configuration settings and used client-side cookies to link client sessions.

To assist with some analysis, we had access to Quova's geolocation database [20], the industry standard for IP geolocation. This database includes 31,080 distinct latitude/longitude locations for more than 6.85 million distinct IP prefixes. When we later refer to calculating the geographic distance between two IP addresses, we refer to first performing longest-prefix match to find the most appropriate IP prefix in this geolocation database, then computing the geographic distance between the latitude/longitude coordinates for these two prefixes.

As Table 2 shows, our dataset includes hosts from approximately two-thirds of all autonomous systems and more than one-third of all BGP-announced IP prefixes (per RouteViews [25]). It covers more than one-half of Quova's identified locations and 214 of 231 countries.

3.4 Summary of results

This section summarizes our two main results:

NATs and DHCP do not contribute largely to edge opacity from the viewpoint of a server. While the majority of clients are behind NATs (roughly 60% from our measurements), virtually all NATs are small. In fact, NAT sizes tend to follow an exponential, rather than a power law, distribution (§4.2). Second, IP reallocation due to DHCP is slow, generally on the order of days. For example, fewer than 1% of the clients we measured used more than one IP address to access our servers over the course of one month, and fewer than 0.07% of clients used more than three IP addresses (§4.3).

Proxies pose a greater problem for IP-based server decisions. While only 15% of clients we measured traversed proxies, these proxies were generally larger than NATs and often served a geographically-diverse client population (§4.4). This latter property is especially important if access control decisions are being made for regulatory compliance. Additionally, as a single client may

```
<iframe src='http://www.cdn.coralcdn.org/noredirect.html?teamid=fcc9c...0c7'
        scrolling=no frameborder=0 marginwidth=0
        marginheight=0 width=1 height=1></iframe>
```

Figure 1: Our measurement web-beacon

use many proxies in multiple locations, blacklisting proxy IP addresses in such cases is largely ineffective.

From these results, we conclude that a principal concern for server operators is the ability to detect and react to proxies. In §5, we introduce a set of classifiers with which a server, using only a single client request, can detect proxies with high accuracy.

4 The Extent of Edge Opacity

4.1 Establishing a comparison set

The analysis in the next two sections requires the ability to detect the existence of a middlebox, as well as to determine whether the middlebox is a NAT or proxy.

To do so, we first build a set of all measured hosts that have clear identifiers of having come through a NAT or a proxy. We characterized a client as having traversed a middlebox if its public IP address differed from the local IP addresses, as returned by our Java applet. (We note that those clients whose local and public IP address are the same were classified as *non-middleboxes*, a dataset used later in §5.1).[4]

We also built an independent comparison set consisting solely of proxies and another consisting solely of NATs. An IP address was considered a proxy if its request contained a standard proxy header, *e.g.*, `Via`. We classified a middlebox as a NAT iff (1) the SYN fingerprint was not a known proxy type, *e.g.*, Cisco or NetApp, (2) the request did not contain proxy headers, (3) the domain name of the middlebox was not suggestive of a proxy, *e.g.*, contained *proxy*, *prx*, *cache*, or *dmz*, and finally, (4) the ratio of distinct `User-Agent` strings to distinct SYN fingerprints was at most three (we explain the rational for this metric in §5.2.2). Under these criteria, we classified 128,066 public IPs as proxies and 444,544 as NATs.

We will also use these sets of known proxies and non-middleboxes in Section 5, in order to evaluate the accuracy of a number of lighter-weight proxy detection heuristics that do not rely on Java.

4.2 NAT characteristics

We begin by showing that the majority of NAT'd networks only consist of a few hosts. To establish this result, we leverage the fact that most devices assign addresses from their DHCP address pool in linear order, starting from some (usually well-known) default value. For example, the DHCP pool of Linksys routers starts at 192.168.1.100 [23]. Then, given the local IP addresses of our clients, we analyze the frequency histogram of the private address space usage between NATs.

Given the frequency histogram of the private address range, we have found that one can easily detect the default starting addresses used by DHCP servers. For example, Figure 2 shows a histogram of 192.168.0/24: There are two clear spikes starting at 192.168.0.2 and 192.168.0.100. Thus, given a common starting value (the beginning of a spike) and given that IP addresses are largely assigned in linear order, we can estimate the distribution of NAT'd network sizes that share the same starting IP address.

More specifically, a client's rank in a group of size n can be approximated by its IP address subtracted by the default starting address. We term this value a host's *IP rank*. In the limit, the average IP rank of measured hosts behind a NAT will converge to $\frac{n}{2}$. Given a sufficient sampling of hosts from a single public IP address, this could be used to approximate the NAT's size. Of course, this technique is useful only for a *particular* NAT and requires measurements from multiple hosts behind that NAT. Our dataset contained requests from multiple hosts in fewer than 10% of the NAT'd public IPs.

However, one can still reach useful conclusions given aggregate data across many NAT'd IPs. Recall that Figure 2 plots an aggregate histogram of the frequency with which each private IP address is used by unique hosts. (again, unique hosts are identified by a unique *(pub-ip, local-ip, SYN-FP)* tuple.) We can use this aggregate information to determine the relative sizes of individual NAT'd networks. To do so, we take the discrete derivative of the histogram, which approximates the ratios of relative NAT sizes. As a simple example, a derivative value of ten for IP rank 1 and derivative value of two for IP rank 2 implies that NAT'd networks with one unique host are five times more likely than networks with two hosts.

Figure 4 shows the discrete derivative calculated over the smoothed histogram[5] for IP address range 192.168.1.100 through 192.168.1.199, inclusive. We use this range for demonstrative purposes, as it contains a single clear spike (see Figure 2): this implies that most hosts within this range use the same starting IP address within their DHCP pool, *e.g.*, due to Linksys routers' default con-

[4] During our analysis, we uncovered many *transparent* web proxies which do not perform address translation. However, since these do not render the client's IP address opaque, we treated them in the same manner as client requests which did not traverse a middlebox.

[5] Using a degree-five Bernstein basis polynomial for smoothing [24].

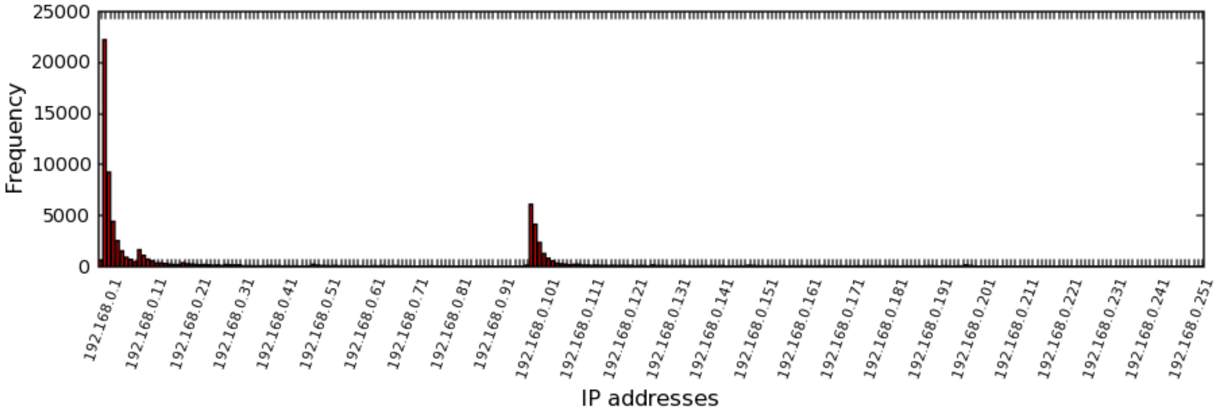

Figure 2: Histogram of the IP address usage within the 192.168.0/24 prefix of all clients which ran the Java Applet

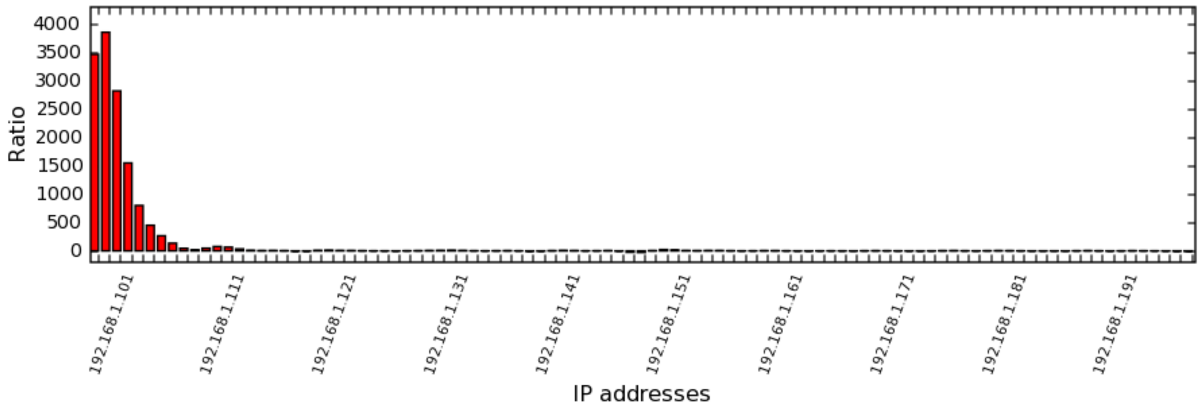

Figure 3: Discrete derivative of the smoothed frequency histogram for 192.168.1.1xx

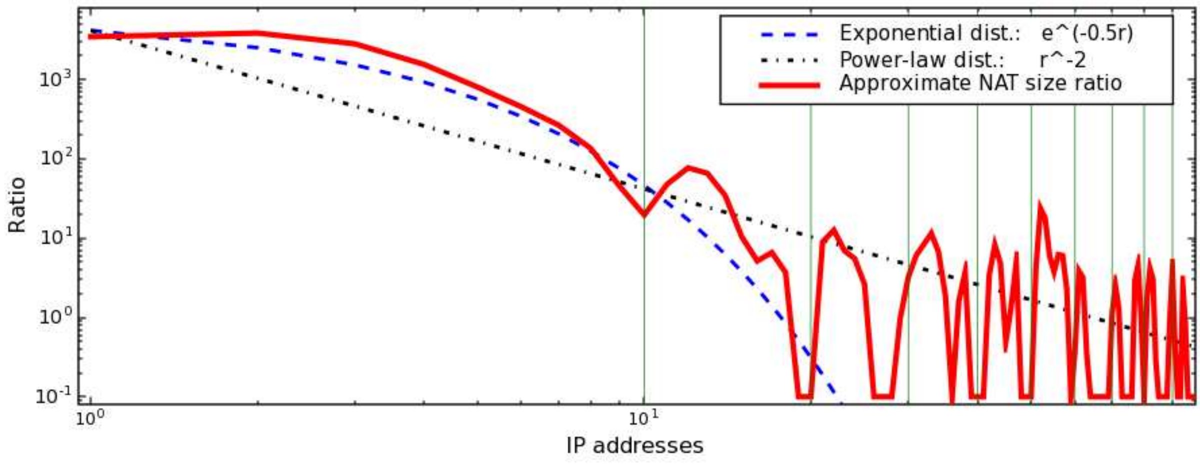

Figure 4: Discrete derivative of the smoothed frequency histogram for 192.168.1.1xx on a log-log scale, with common exponential and power-law curves plotted as comparison. Vertical lines are drawn every 10 IP addresses.

figurations [23]. With the derivative peaking at IP rank 2, we conclude that for this IP range, the most common NAT'd network has two hosts. In fact, the vast majority of networks in this range have fewer than 7 hosts, and there are almost 200 times fewer networks of 10 hosts than those of 2 hosts (IP rank 10 has value 19.6).

Figure 4 plots the same discrete derivative on a log-log scale, given as the solid line. We see that the curve is somewhat periodic, with new (yet smaller) peaks roughly correlating to intervals of ten IP addresses. While points at increasing IP rank may be caused by the existence of (outlying) large networks, this periodicity likely indicates less-common defaults used by some DHCP pools for numbering a network.

Figure 4 also includes an exponential and a power-law distribution curve that attempt to fit the data. The plot suggests that NAT'd networks follow an exponential distribution: The first curve of IP address usage starting at 192.168.1.100—before the periodic jump at 192.168.1.110—closely follows $e^{\frac{-r}{2}}$.

We applied this NAT-size analysis to all other populated regions of the private address space that we measured, including 10/8, 192.168/16, 172.16/12, and 169.254/16. Overall, the characteristics of the analyzed IP ranges appeared similar to the results presented above. One exception is that 10/8 had (only slightly) largely NAT sizes, yet we also found an order of magnitude fewer hosts using 10/8 networks compared to 192.168/16 networks. We do not include the graphs of these results due to length limitations.

Analyzing the data less in aggregate, of the 444,544 NAT'd networks we measured, only 112 networks have more than 10 hosts (or $\leq 0.03\%$). The largest NAT'd networks we discovered were heavily biased based on geographic location: Of the largest ten networks, eight were residential ISPs in Japan (including all top-five), with one additional ISP in China and one for Google. As we show in Section 5, it is possible for a server to detect and track exceptionally large NATs.

4.3 DHCP usage characteristics

In this section, we analyze dynamic IP renumbering. To do this, we track clients using HTTP cookies. All results presented in this section were derived from a one month subset of our data in which we added javascript to perform cookie management. The dataset covers 537,790 individual clients (where clients are identified by unique cookies) across 4 million web sessions.

Figure 5 shows the extent to which clients use multiple IP addresses over time. Each histogram bar for time epoch i includes those clients whose first and last accesses to our servers are fully contained within the time period $[i, i+1)$. Each bar additionally shows the breakdown of the number of IP addresses used by its respective client

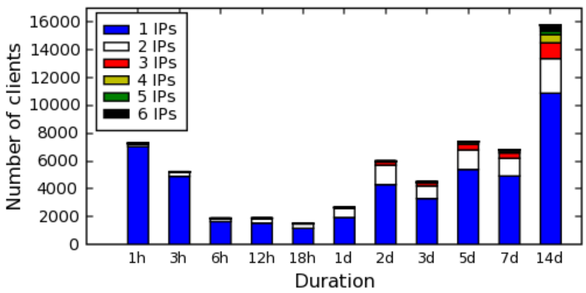

Figure 5: Prevalence of multiple IP usage by clients over time

Clients affected	All public IPs	Proxy IPs only
2	9.7%	53.4%
3	5.1%	40.2%
5	3.1%	28.3%
10	1.4%	17.3%

Table 3: Probability of collateral damage when blacklisting public IP addresses.

over its lifetime. We excluded only those clients who use more than one public IP address within 10 minutes—hand verification shows such clients to be using some form of load-balancing or anonymizing proxies.

We find that fewer than 2% of clients use more than 2 IP addresses between 3-7 days, with only 8% of clients using more than 3 IPs in 2-4 weeks. In fact, 72% of nodes used a single IP address (and 91% used 2 IPs) to access our servers for up to 2 weeks! Thus, we conclude that DHCP allocation results in little IP renumbering of clients from a server's perspective.

If clients switch IP addresses relatively rarely, can a server simply blacklist an IP address and not worry about the potential collateral damage to other hosts that may be effected by this filtering rule? In other words, how often would other clients re-use the same IP address already blacklisted?

Table 3 shows the extent of this collateral damage over the course of the dataset's month duration. We find that over 90% of the time, blacklisting any IP address would not have effected any other clients. Only in 1.4% of the cases would it effect more than 10 clients. In fact, the damage may even be less, as this analysis uses cookies as a form of identity. Clients may delete their cookies and thus inflate the measured damage. Indeed, other metrics of client identity—such as generating a client fingerprint based on a large set of browser configuration settings, many of which have significant entropy—appear to yield better preliminary results, although we do not include such analysis due to space limitations.

Finally, Table 3 demonstrates the importance of handling proxies differently from NAT'd and non-NAT'd IP addresses. We see that blacklisting a proxy would have some collateral damage 53% of the time and impact more than 10 clients in over 17% of cases.

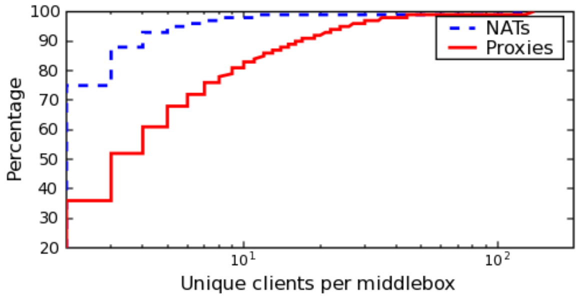

Figure 6: CDFs of the number of unique hosts behind each NAT and proxy devices

4.4 Proxy characteristics

This section shows that proxies both tend to be larger than NATs and often serve clients with high geographic diversity. These results are unsurprising: while virtually all home routers support NAT, proxies are often deployed in support of larger organizations. Further, unlike most NATs (VPNs excepted), proxies do not need to be on route and thus can serve clients from anywhere on the Internet.

Per §4.1, we created a set of *known proxies* by including the public IP addresses of those clients whose requests contained popular proxy headers. All IPs characterized as proxies also must have had a least one client that ran Java to reveal the existence of address translation.

Proxy sizes. Many of the larger proxies' clients contain publicly-routable local addresses. Unfortunately, this makes it difficult to estimate proxy sizes using the same DHCP allocation insight from §4.2. Instead, we plot the number of unique clients which accessed our system through proxies versus those using NATs (Figure 6). This figure only includes middleboxes in which we identified two or more distinct clients.

The graph shows that, as viewed from our servers, proxies serve a larger client population than do NATs. For example, only 10% of NATs were traversed by three or more hosts, opposed to 50% of proxies. While these results may not be representative for all servers, they do align with our intuition that proxies are likely to represent larger client groups.

Client-proxy locality. We now explore the proximity between proxies and the clients they serve. This subsection's results are limited to those pairs for which the proxy's public IP address differs from the publicly-routable local IP address of the client. Our dataset includes 26,427 such pairs of clients and web proxies. All geographic location information is obtained from Quova[20].

Because proxies are often off-route, they have two characteristics not found in NATs. First, clients from globally-reachable proxies can originate from anywhere. Figure 7 demonstrates this by mapping out all clients accessing our measurement servers through a Google web proxy located in Mountain View, CA. While a server operator may be comfortable whitelisting an IP shared by a large, albeit localized, NAT, she may wish to restrict unfettered access to such a proxy.

Figure 7: Location of clients accessing web via Google proxy (identified by marker)

Figure 8: Location of anonymizing egress points used by clients within a single class-C network (identified by marker)

Secondly, while NATs virtually always map a small set of globally-reachable IP addresses to a larger number of machines, proxies are sometimes used to give a client access to multiple public IP addresses. For example, Figure 8 plots the actual location of colocated clients connecting to our measurement site through numerous proxies around the world. We believe these proxies to be part of an anonymization network. In such cases, where a single machine commands multiple geographically- and topologically-diverse IP addresses, attempting to curtail a particular client's access through IP blacklisting is near impossible.

From a more quantitative perspective, Figure 9 shows a CDF of the distance between clients and their web proxies. 51.4% of clients have the same location as their proxies. However, fully 10% of client-proxy pairs are located more than 3,677 kilometers apart, while 1% are located more than 10,209 kilometers apart. Table 4 summarizes locality properties with respect to countries and regions for our dataset. We also analyzed clients' logical proximity to the proxies they traverse. We found that only 29.2% of clients shared the same AS number as their proxy and only 6.7% shared the same IP prefix (per RouteViews BGP data [25]).

Finally, for completeness, we analyzed the routing inefficiency caused by clients' detour routes to their web

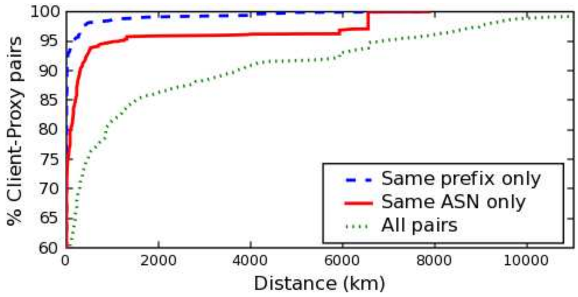

Figure 9: Proximity of web proxies to their clients

Country	City	Geolocation	SLD
85.1%	51.4%	51.4%	56.1%

Table 4: Percentage of client-proxy pairs located within the same country, city, or distinct location, or belonging to the same second-level domain (*e.g.*, *stanford.edu*).

proxies (Table 5). All destinations considered in this dataset correspond to our measurement servers, all located in Stanford, CA.

While these results do not directly impact edge opacity, it does provide some insight into the degree with which clients go out of their way to access content. The choice of longer detour routes may be forced on customers by ISPs that choose to deploy fewer sets of proxies, *e.g.*, for easier management. For example, much of AOL's European web traffic exits its network at proxies located in Frankfurt, Germany [1]. On the other hand, longer detour routes may be chosen by clients purposefully, either for anonymization or to skirt regulatory-compliance restrictions. In fact, we were able to detect numerous instances of clients using anonymizers (similar to that in Figure 8), making them normally appear as multiple clients distributed world-wide.

5 Middlebox Detection

We now present and analyze a set of techniques for detecting the usage of NATs and proxies. We first describe a toolkit of classifiers which can be used to detect proxies in real-time. We then describe how a server can use client history to determine the IP addresses of large NATs, as well as prefixes with high rates of IP renumbering among its clients. Finally, we describe our implementation of a custom web-server that implements such techniques.

5.1 Real-time proxy detection

The previous section's results suggest that proxies have a higher potential of negatively impacting server decisions. Not only are their client populations geographically distributed, but, unlike with NATs, a single client can hide behind multiple proxies. Fortunately, we find that it is

	50%	75%	90%	99%
Direct distance (km)	3890	8643	10611	16908
Detour distance (km)	4135	9156	13038	18587
Stretch	1.06	1.06	1.23	1.10

Table 5: Comparing direct vs. detour distances of web proxies

possible for a server to detect proxies in real time, *i.e.*, immediately following a single client request.

To do so, we introduce and analyze the following set of proxy indicators: (1) checking HTTP headers, (2) SYN fingerprinting clients, (3) geolocating clients, (4) comparing client timezone against geolocation information, (5) comparing client language against geolocation, and (6) analyzing application versus network round-trip-times. In what follows, we analyze the efficacy of each technique as well as their ability to detect proxies in the aggregate. Our analysis uses client configuration state returned by active content, passive server-side analysis, limited active probing, and IP geolocation (specifically, Quova's package [20]).

It is also possible to detect proxies using more intrusive techniques. One common method is to attempt to connect back to an incoming client's IP address at known proxy ports. Obviously, this technique only works for public proxies. Another approach is to use a client-side Java applet that issues a GET request for a popular object from a different domain than the server, *e.g.*, the banner image for `google.com`. A successful request would imply a cache hit on an on-path proxy. In this section, however, we limit our evaluation to the former six approaches, as they are both light-weight and have low visibility.

To evaluate the utility of each heuristic, we test them against the 22,546 known proxies per §4.1, as well as those clients classified as non-NAT'd hosts (non-proxies). A false positive (FP) occurs when a technique classifies a non-proxy as a proxy, while a false negative (FN) corresponds to a proxy being classified as a non-proxy. We summarize the results of each classifier for this *headers* dataset in Table 6.

Table 6 also evaluates our classifiers against an alternative *dns* dataset of proxies: The set of middleboxes having a domain name suggestive of a web proxy (*e.g.*, containing *cache*). While this domain-name test does not provide the certainty offered by proxy headers, it provides another useful testing set.

Overall, we find that our combined classifiers can detect a proxy between 75% and 97% of the time, with a false positive rate between 1.9% and 2.5%. These numbers merely serve to qualify the proposed classifier's effectiveness, as we combined them in a rather straight-forward manner. Certainly, a more intelligent weighting of individual classifiers' results (perhaps using machine learning techniques) should be able to improve the combined classifier's accuracy.

Classifier	ID'd proxy		False Negatives		ID'd non-proxy		False Positives	
dataset	headers	DNS	headers	DNS	headers	DNS	headers	DNS
Headers	n/a	94.9	n/a	5.1	n/a	100.0	n/a	0.0
SYN-FP	67.3	65.3	5.8	2.4	86.8	86.3	1.9	1.9
Geolocate	18.0	24.4	n/a	n/a	n/a	n/a	n/a	n/a
Timezone	6.1	5.0	n/a	n/a	n/a	n/a	2.0	2.0
Language	5.7	4.8	n/a	n/a	n/a	n/a	0.5	0.6
RTT	52.9	?	n/a	n/a	n/a	n/a	9.1	?
Combined	75.7	97.6	5.0	2.3	85.0	97.5	2.5	1.9

Table 6: Effectiveness of real-time classifiers for known proxies (in percentages) for both the header- and dns-based proxy datasets. *n/a* corresponds to a test being non-applicable; ? denotes insufficient data.

Proxy headers. Web proxies are required by the HTTP/1.1 specification (RFC 2616 [14]) to add their information to a `Via` header for both HTTP requests and responses that they proxy. Unfortunately, not all proxies do such, making proxy detection a non-trivial task.

Limiting our consideration to the *known-proxy* dataset—which by construction must include at least one proxy header—13.5% lack the required `Via` header. 69.6% of these proxies include an `X-Forwarded-For` header (a non-standard header introduced by Squid [30]) and 9.0% include `X-Bluecoat-Via` [6]. Although we tested for five other anecdotally-used headers (*e.g.*, `Client-IP`), we did not found any evidence of other such proxy headers in use.

On the other hand, if we consider the set of proxies in the *dns* dataset, we find that 94.9% include at least one proxy header. Of course, given their indicative hostnames, these proxies are not attempting to hide their identity, unlike many anonymizing proxies.

Client SYN fingerprinting. Our modified web server captures the SYN packet of all incoming requests and uses it to generate the client's SYN fingerprint [27]. SYN fingerprints (SYN-FP) provide an estimate of the sender's operating system. In the case of web proxies, which terminate their clients' TCP sessions, the SYN fingerprint corresponds to the proxy's TCP stack, not that of the clients. SYN fingerprints can be immediately used to uncover a variety of operating systems not commonly belonging to end-hosts (*e.g.*, Cisco, NetApp, and Tru64).

We can increase the coverage of this classifier, however, when combining SYN-FP information with operating system information returned in a client's HTTP `User-Agent` string. Specifically, we flag a host as a proxy if its SYN-FP host-type differs sufficiently from its `User-Agent` host-type, and mark it as a non-proxy if they are the same (although this latter step can certainly lead to false negatives).

Analyzed against our known proxies and non-proxies, this classifier successfully identified 67.3% proxies and 86.4% non-proxies, with a 2.3% false positive (FP) and 5.8% false negative (FN) rate. The classifier failed whenever the SYN-fingerprint did not match well-known operating-system characteristics. The higher FN rate is largely due to vagaries in the Windows TCP stack, yielding sometimes-inaccurate host-type estimations (*e.g.*, NT 5.0 vs. NT 5.1) from SYN fingerprinting.

Geolocation. Given a client's local IP address (from our Java applet) and its public IP address, we directly compare the geolocation information known about these two IP addresses. This method is limited to clients whose local IPs are globally reachable. But as Table 4 shows, these locations often differ. When classifying an IP address as a proxy if its public and local IP addresses differ in location, we are able to successfully identify 18.0% of our known proxies. We make no conclusion about those clients sharing the same location as their public IP address—while proxies may serve geographically-diverse clients, they can certainly serve local ones as well—hence the lower identification percentage. Additionally, given that our list of known non-proxies required matching local and public IP addresses, we cannot perform such analysis on non-proxies to generate an FP rate.

Client timezone. We now consider how client geolocation hints may differ from the corresponding information supplied by the server's geolocation package. This approach is unnecessary if a client can be geolocated directly (as described immediately above), but it provides a lighter-weight approach and may be applicable when a client's local IP address is in a private prefix.

Specifically, a web client exposes its timezone information in its `Date` header (as well as directly via javascript). Our timezone classifier compares client timezone information to that of its geolocated public IP address, over all client requests arising from that IP address. Unfortunately, this approach does not appear very useful, identifying only 6.1% of known proxies while yielding a 2.0% FP rate, given the low apparent rate of requests across timezones as compared to misconfigured or mobile hosts. Note that direct IP geolocation (as above) using a commercial database provides much higher granularity and thus higher accuracy.

Client language. Similar to timezone information, we see how client language information may serve as a geolo-

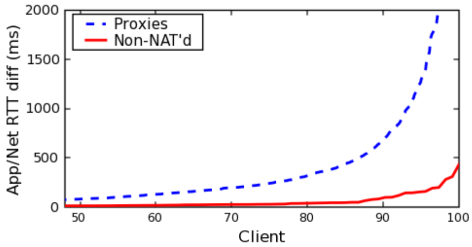

Figure 10: Differences in application- versus network-level RTT measurements between proxies and non-proxies

cation hint. Client language is collected via javascript, although it is also available per the `Accept-Languages` HTTP header. After building a database that maps languages to countries, we asked whether a client's IP address is located within a country using that language. If some sizable fraction of an IP address's clients cannot be matched via language, the IP address is marked as a proxy.

This language heuristic identified 5.7% of our known proxies, with a 0.5% FP rate. The coverage of this classifier is significantly limited by the universality of some languages—*e.g.*, browsers and hosts are configured with language settings of `en-us` world-wide, which we subsequently ignored—which led to our inability to make meaningful decisions on a majority of clients.

RTT analysis. Finally, we use differences in application-level and network-level round-trip-times (RTTs) to detect proxies. Specifically, we compare the difference between TCP RTT measurements on the server-side to the minimum request time over multiple HTTP GET requests issued by the client. We use javascript to perform the application-level RTT measurements (by timing asynchronous `XMLHttpRequests`).

A large difference between application- and network-level RTTs may suggest a client using a distant proxy. Figure 10 compares this difference as measured against our known proxies and non-proxies. While non-proxies can show a significant RTT gap—likely due to application-layer scheduling delays yielding inaccurate RTT measurements, even across multiple HTTP requests—this RTT gap is much more pronounced among clients of proxies.

This technique was able to identify 52.9% of our known proxies (using an RTT difference threshold of 50ms), although it had a much higher FP rate at 9.1% due to non-proxies experiencing significant application RTT delays. We draw no conclusion if a client has a similar network and application-level RTT.

Additionally, unlike some previous techniques which used client-supplied configuration state, this approach is less susceptible to malicious clients: a client that does not have control over the proxy it traverses can only make itself appear further from its proxy—and hence more likely be flagged as indeed behind a proxy—not closer.

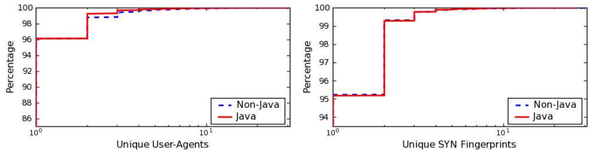

Figure 11: CDFs of unique User-Agent strings and SYN-FPs per public IP for hosts that did and did not run the Java applet

5.2 History-based middlebox detection

This section presents a methodology by which servers can identify large NATs and proxies, as well as distinguish between the two, by using the history of requests seen from a particular IP address or prefix. For this analysis, the server must record the public IP, SYN-FP, and `User-Agent` for each request.

We first show that non-NAT'd hosts show little variability in User-Agent strings and SYN fingerprints, as compared to clients behind middleboxes. Coupled with cookie usage, this heuristic provides a strong first-order differentiator of NAT'd and non-NAT'd hosts. Second, given that non-NAT'd hosts show little variability in these parameters, one can often differentiate between individual clients behind a middlebox using User-Agents alone, as the strings have a large amount of entropy. Finally, we determine the accuracy of detecting and differentiating between NAT'd and proxied networks by analyzing the distribution of User-Agents and SYN-FPs.

As before, we limit our study to those clients that ran the Java applet. However, as Figure 11 shows, the User-Agent and SYN-FP uniqueness characteristics of the entire dataset closely matches those of the subset of clients that ran the applet. Thus, we believe that our results generalize to the larger dataset.

5.2.1 Identifying NATs and proxies

How do the characteristics of non-NAT'd hosts differ from those of NAT'd hosts? One obvious approach to detect the presence of multiple clients behind a middlebox is to monitor a public IP address for any changes in its local hosts' configurations (*e.g.*, suggesting different operating systems or application versions). This approach is commonly discounted, however, due to dynamic IP addressing, routine changes to machine configurations, operating system or application updates, and even physical hosts that support multiple operating systems.

However, we have found that hosts identified as non-NATs—*i.e.*, hosts having the same local and public IP address—have a relatively low variability in User-Agents and SYN-FPs over time. Figure 12 shows the number of unique User-Agents for non-NAT'd hosts compared to those behind middleboxes. For the hosts behind middleboxes, we only include public IP addresses from which we identified two or more local IP addresses.

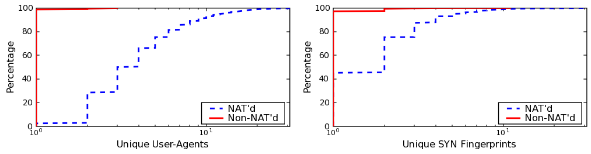

Figure 12: CDFs of unique User-Agents and SYN-FPs per public IP addresses for non-NAT'd hosts and NAT'd hosts

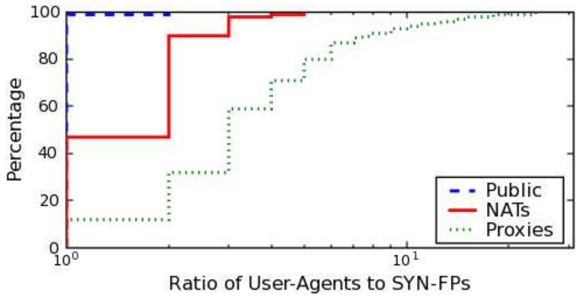

Figure 13: User-Agent to SYN-FP ratio for public (non-NAT'd), NAT'd, and proxied hosts

Fewer than 1% of the non-NAT'd hosts have multiple User-Agents, while the most any single host presented was three. Additionally, only 2.5% of such hosts had more than one SYN-FP, with a maximum of 11. While this does not show that non-NAT'd hosts are not being periodically renumbered via DHCP, it does corroborate our results in §4.3 that these rotations are sufficiently slow from a server operator's vantage. On the other hand, requests from middleboxes serving at least two clients had a much higher number of unique User-Agents and SYN-FPs.

This lack of different User-Agents from non-NAT'd addresses is not due to their paucity, however, as we found a total of 164,122 unique User-Agents among all clients. In fact, the frequency plot of these User-Agent strings follows a clear power-law distribution (we omit due to space limitations). This distribution suggests that there exists sufficient entropy to often differentiate between individual hosts' User-Agents.

5.2.2 Differentiating NATs and proxies

Given that the variability in User-Agents is a good first-order detector of middleboxes, we now turn to determining whether a middlebox is a NAT or proxy. Certainly, all the real-time methods introduced in §5.1 can be used to detect proxies. However, we also find that comparing the ratio of User-Agents to SYN-FPs for a particular IP address over multiple requests is useful for distinguishing between non-NAT'd, NAT'd, and proxied hosts.

Figure 13 plots the ratio of User-Agents to SYN-FPs for these three types of hosts (non-NAT'd, NAT'd and proxies). As before, we only include middleboxes from which we observed at least two clients. One finds a clear distinction between these three classes. For non-NAT'd hosts, the ratio is one in 99.99% of the cases — that is, in almost all cases, there is exactly one user-agent string per unique SYN-FP per IP. If the ratio is ≥ 3—that is, there are at least three distinct user-agent strings per SYN-FP—the middlebox is a proxy with high probability.[6]

5.3 Implementation

We have implemented the techniques discussed in this section into a custom web server which provides real-time proxy and NAT detection to aid in access-control decisions or other IP analytics applications. Our implementation consists of approximately 5,000 lines of C++, as well as the accompanying javascript and Java applets.

Figure 14 shows how our server, hereafter called the *illuminati-server*, integrates into an existing website. The website includes both a standard Web server (serving dynamic pages), an IP analytics engine, and the illuminati-server. In fact, we are currently integrating and deploying the illuminati-server alongside Quova's GeoDirectory Server [20] (the analytics engine in Figure 14) for commercial real-time proxy detection and geolocation.

A website integrates this solution as follows. First, the webserver dispatches client requests to the illuminati-server by adding an embedded `iframe` (similar to Figure 1) or `script` object to any pages for which it wishes to gather IP-based information (Step 1). This embedded object returned by the web server is tagged with a unique session id with which to later identify a particular client (as multiple clients may share the same public IP address). The client then fetches the embedded object from the illuminati-server, which subsequently serves an appropriate javascript or Java applet to the client (Step 2). This object is loaded in the background and therefore does not contribute to client-perceived latency on modern browsers.[7] After executing, this active content sends its client-generated information back to the illuminati-server, along with the server-specified session identifier (Step 3). Next, the illuminati-server pushes this information to the IP analytics engine (Step 4), which both stores them for historical analysis and, combining the data with IP geolocation information, analyzes whether the client is behind a NAT or proxy and, if the latter, where the client might be actually located. Depending on the results of this analysis, the IP analytics engine might call for other code to be executed by the client (*i.e.*, repeat Steps 2-4). Finally, the results of this analysis can be queried by the decision logic of the website, keyed by session identifier, at any point after it is collected (Step 5).

In this particular configuration, the time between a client's initial request and the time that a detection result

[6] We hand-inspected all middleboxes that the above criteria identified as NATs yet displayed a high User-Agent-to-SYN-FP ratio. From this, we identified 30 more large proxies missed using our initial criteria.

[7] Unless the Java virtual machine must be booted, which can cause some noticeable delay.

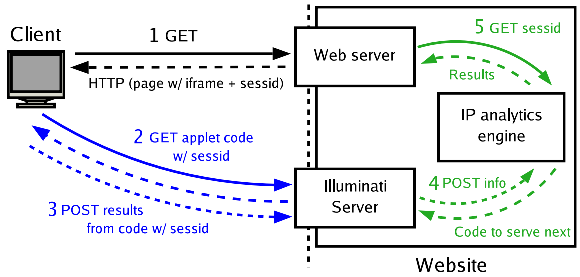

Figure 14: Integrating the illuminati server

may be available is at least 2 RTTs: half an RTT to return the initial page, 1 RTT to get the embedded code, and half an RTT to post the results. Of course, if application-level RTT tests are also used, each measurement adds one additional RTT. Serving additional applets increases this delay as well. Still, we expect that proxy detection would primarily be used for pages without strict latency requirements, such as to accompany user authentication.

Notice that this implementation provides for a separation of function between a simple stand-alone illuminati server and the more complex IP analytics engine. The illuminati-server must be deployed outside of any local reverse proxy, so that it receives direct TCP connections from a client (in order to get the client's public IP address, SYN headers for fingerprinting, and network RTT times). The illuminati-server and engine can also be hosted at some site external to the webserver, in which case the webserver would remotely query the engine over a secure channel. In both cases, the service can be integrated easily into existing websites.

6 Related Work

Architectural proposals. Many research proposals have offered new protocols or architectural changes which support globally identifiable end-hosts. These include HIP [22], DOA [33], LNA [2], UIP [15], IPNL [16]. TRIAD [8] and i3 [31]. These proposals generally focus on solving multiple architectural problems including edge opacity. One of the primary motivations is to decouple end-host identifiers from topology, allowing end-hosts to retain the same identity during mobility and multi-homing. However, globally unique end-host identifiers could also provide a basis for access-control decisions and thus ameliorate the ills caused by NATs.

Unfortunately, even if these architectures were to be adopted, proxies may continue to mask client identities. Proxies generally operate at the session layer, yet most of these proposals suggest adding a shim header between the IP and transport layer. Our findings suggest proxies are widely used for load balancing, caching, anonymizing, and skirting access controls. It is unclear how mechanisms operating below the transport layer address the issue of proxies.

IPv6 [11] confronts the problem of end-to-end client reachability by using 128-bit addresses. IPv6 also mandates support of IPSec, which, with the use of the AH header, makes middlebox deployment difficult. However, we believe that even with widespread deployment of IPv6, NATs and proxies will continue to persist.

With NUTSS [21], Guha *et al.* focus on the problem of end-to-end addressability through NATs using SIP [28] (for signaling and long-term, stable identifiers) and STUN [29] (for NAT punching). While this approach has made substantial inroads for peer-to-peer applications such as VoIP, it is not widely used in the client/server paradigm such as the web, in which clients are largely anonymous.

NAT and proxy measurement. There are few reliable methods for detecting unique hosts behind NATs and proxies. Bellovin presents a technique using the `IPid` field in [4]. This approach relies on the host operating system using unique yet predictable `IPids`. To be effective, however, traffic from multiple clients behind the same NAT must be present within a short time period: This latter constraint makes this approach impractical for any but the most popular Internet systems. For example, fewer than 5% of the NATs sent multiple hosts to our collection site over the full 7 month measurement period. This technique is also ineffective for detecting proxies.

Another approach seeks to identify NATs themselves by examining packets for specific modifications that occur as packets traverse some types of layer-3 NATs [27]. However, this technique can not identify the number of active clients behind the NAT. Thus, such a classification is not useful for IP-based client identification: as we have shown in §4.2, while 74% of our clients are located behind NATs, the majority of NAT'd networks are only comprised of a few hosts (often just one).

Ganjam and Zhang [19] present measurements of NAT usage from a deployment of a video broadcast application. They observed environments where the percentage of NATs ranged between 35% and 80%. Their data set covered 553 hosts.

Effects of dynamic IP renumbering. IP aliasing, where a unique client is allocated multiple IP addresses via DHCP over time, has the potential to hinder a server's ability to detect unique clients. Bhagwan *et al.* [5] studied aliasing in Overnet, a peer-to-peer file sharing system. They probed 1,468 unique hosts over a period of 7 days and found that almost 40% used more than one IP address. We show that, from a server's viewpoint, aliasing does not seriously effect its ability to differentiate clients nor would result in meaningful collateral damage given IP-based blacklisting.

7 Conclusions

Conventional wisdom abounds with the drawbacks of using IP-based identification in the face of NATs, proxies, and dynamic IP renumbering. Blacklisting large proxies or NATs can result in legitimate clients losing access to desired services, while whitelisting can give access to unwanted clients. The actual extent of the problem, however, has remained largely a mystery.

Part of the challenge in uncovering the impact of edge opacity is a lack of practical techniques and deployments to "peer through the shroud" of middleboxes in the wild and at scale. In this work, we have developed, deployed, and analyzed a set of techniques that enable such measurement. Our approach combines active content—downloaded and executed by unmodified web clients—with server-side measurements.

We present seven months of measurement results from nearly 7 million clients across 214 countries. Our results show that while 74% of clients are behind NATs or proxies, most NATs are small and follow an exponential distribution. Furthermore, the few exceptional large NATs we found were easy to characterize from the server's perspective. Dynamic renumbering from DHCP generally happens on the order of days. Indeed, fewer than 2% of the clients that visited our servers over a week's period used more than two IP addresses.

Proxies, on the other hand, service client populations that may be both larger and geographically diverse. Thus, poor access control policies for proxies can have greater negative implications. However, we show that a server can detect a proxy both in real-time and using historical analysis, thus enabling a server operator to make more informed decisions.

That said, this paper's analysis is only a first step towards understanding the extent and impact of edge opacity on server decisions. We have focused on how edge opacity affects access control decisions declared over IP addresses. However, as alluded to earlier, characterizing edge opacity has implications to other domains, including regulatory compliance and fraud detection. To this end, we are currently integrating our implementation into Quova's widely-deployed IP intelligence engine [20] and plan to explore its utility in these areas.

Acknowledgments. We are very grateful to the illuminati community for contributing data by adding our web-beacons to their sites. Quova, Inc. provided access to their IP geolocation database and worked with us to develop and deploy an implementation of this work. Jeffrey Spehar contributed to our analytics website, and David Andersen, Nick Feamster, David Mazières, Dan Wendlandt, our anonymous reviewers, and our shepherd, Rebecca Isaacs, provided feedback on earlier versions of this paper.

Finally, we wish to acknowledge the support of our advisors, David Mazières and Nick McKeown.

References

[1] AOL. Transit data network. http://www.atdn.net/, 2006.

[2] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the internet. In *SIGCOMM*, Sept. 2004.

[3] BBC News. Bush website blocked outside US. http://news.bbc.co.uk/2/hi/technology/3958665.stm, 2004.

[4] S. M. Bellovin. A technique for counting NATted hosts. In *Internet Measurement Workshop*, Nov. 2002.

[5] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *IPTPS*, Feb. 2003.

[6] Bluecoat. Blue Coat Systems. http://www.bluecoat.com/, 2006.

[7] M. Casado, T. Garfinkel, W. Cui, V. Paxson, and S. Savage. Opportunistic measurement: Extracting insight from spurious traffic. In *HotNets*, Nov. 2005.

[8] D. Cheriton and M. Gritter. TRIAD: A new next generation internet architecture. http://www-dsg.stanford.edu/triad/, Mar. 2000.

[9] CoralCDN. http://www.coralcdn.org/, 2006.

[10] Cyota. http://www.rsasecurity.com/node.asp?id=3017, 2006.

[11] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, Dec. 1998.

[12] Digital Envoy. http://www.digitalenvoy.net/, 2006.

[13] DIMES. http://www.netdimes.org/, 2006.

[14] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. 1999.

[15] B. Ford. Unmanaged internet protocol: Taming the edge network management crisis. In *HotNets*, Nov. 2003.

[16] P. Francis and R. Gummadi. IPNL: A NAT-extended internet architecture. In *SIGCOMM*, Aug. 2001.

[17] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *NSDI*, Mar 2004.

[18] M. J. Freedman, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for any service. In *NSDI*, May 2006.

[19] A. Ganjam and H. Zhang. Connectivity restrictions in overlay multicast. In *NOSSDAV*, June 2004.

[20] GeoDirectory Server. Quova, Inc. http://www.quova.com/, 2006.

[21] S. Guha, Y. Takeda, and P. Francis. NUTSS: a SIP-based approach to UDP and TCP network connectivity. In *FDNA*, Aug. 2004.

[22] HIP. Host Identity Protocol, Internet Draft, 2006.

[23] Linksys. http://www.linksys.com, 2006.

[24] G. G. Lorentz. *Bernstein Polynomials*. U. of Toronto Press, 1953.

[25] D. Meyer. University of Oregon RouteViews Project. http://www.routeviews.org/, 2005.

[26] minFraud. MaxMind, Inc. http://www.maxmind.com/, 2006.

[27] p0f v2. Passive operating system fingerprinting. http://lcamtuf.coredump.cx/p0f.shtml, 2006.

[28] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, June 2002.

[29] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs). RFC 3489, Mar. 2003.

[30] Squid Web Proxy Cache. http://www.squid-cache.org/, 2006.

[31] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *SIGCOMM*, Aug. 2002.

[32] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *EUROCRYPT*, May 2003.

[33] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes no longer considered harmful. In *OSDI*, Dec. 2004.