Mapping the Expansion of Google's Serving Infrastructure*

Technical Report 13-935, University of Southern California, Department of Computer Science

Matt Calder[1], Xun Fan[2], Zi Hu[2], Ethan Katz-Bassett[1], John Heidemann[2], and Ramesh Govindan[1]

[1]University of Southern California — [2]USC/ISI

ABSTRACT

Modern content-distribution networks both provide bulk content and act as "serving infrastructure" for web services in order to reduce user-perceived latency. These serving infrastructures (such as Google's) are now critical to the online economy, making it imperative to understand their size, geographic distribution, and growth strategies. To this end, we develop techniques that enumerate servers in these infrastructures, find their geographic location, and identify the association between clients and servers. While general techniques for server enumeration and geolocation can exhibit large error, our techniques exploit the design and mechanisms of serving infrastructure to improve accuracy. We use the EDNS-client-subnet extension to DNS to measure which clients a service maps to which of its servers. We devise a novel technique that uses this mapping to geolocate servers by combining noisy information about client locations with speed-of-light constraints. We demonstrate that this technique substantially improves geolocation accurate relative to existing approaches. We also cluster servers into physical sites by measuring RTTs and adapting the cluster thresholds dynamically. Google's serving infrastructure has grown dramatically in the last six months, and we use our methods to chart its growth and understand its content serving strategy. We find that Google has almost doubled in size, and that most of the growth has occurred by placing servers in large and small ISPs across the world, not by expanding on Google's backbone.

1. INTRODUCTION

Internet traffic has changed considerably in recent years, as access to content is increasingly governed by *web serving infrastructures*. These consist of decentralized *serving sites* that contain one or more *frontend* servers. Clients of these infrastructures are directed to nearby frontends, which either directly serve content (e.g., as in a content distribution network like Akamai), or use split TCP connections to relay web acccess requests to back-end data centers (e.g., as in Google's serving infrastructure).

Serving infrastructures are motivated by the desire to optimize user-perceived latency [28]. Web service providers invest heavily in building out these infrastructures and they also develop sophisticated mapping algorithms to direct clients to nearby servers. In recent months, as we discuss later, Google's serving infrastructure has nearly doubled in size. Given the increasing economic importance of these serving infrastructures, we believe it is imperative to understand the content serving strategies adopted by large web service providers, especially Google. Specifically, we are interested in the geographic and topological scope of serving infrastructures, their expansion, and how client populations impact build-out of the serving infrastructure.

Several prior studies have explored static snapshots of content-distribution networks [12, 2, 23], often focusing on bulk content delivery infrastructures [12], new mapping methodology [2], or new DNS selection methods [23]. In contrast, our work focuses on the broader class of web serving infrastructures, develops more accurate methods to enumerate and locate frontends and serving sites, and explores how one infrastructure, Google's, grows over six months of active buildout.

The first contribution of this paper is a suite of methods to enumerate frontends, geolocate them, and cluster them into serving sites. Our methods exploit mechanisms used by serving infrastructures to optimize client-perceived latency. To enumerate servers, we use the EDNS-client-subnet prefix extension [8] that some serving infrastructures, including Google, use to more accurately direct clients to nearby servers. Our novel geolocation technique, which we show to be substantially

*Xun Fan, Zi Hu, and John Heidemann are partially supported by the U.S. Department of Homeland Security Science and Technology Directorate, Cyber Security Division, via SPAWAR Systems Center Pacific under Contract No. N66001-13-C-3001. John Heidemann is also partially supported by DHS BAA 11-01-RIKA and Air Force Research Laboratory, Information Directorate under agreement number FA8750-12-2-0344. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of SSC-Pacific.

more accurate than previously proposed approaches, exploits the fact that serving infrastructures employ sophisticated mapping strategies that determine the frontend or serving site nearest to clients. Our technique, called *client-centric geolocation* (CCG), geolocates a server by taking the geographic mean of the (possibly noisy) locations for clients associated with that server, while filtering out clients with bad location information using speed-of-light constraints. We also cluster servers into serving sites, adding dynamic thresholding and RTT-based fingerprinting to current methods. These changes provide enough resolution to distinguish different sites in the same city. These sites represent unique network locations, a view that IP addresses, prefixes, or ASes can obscure.

Our second major contribution is a detailed study of Google's web serving infrastructure, and its recent expansion over the last six months. To our knowledge, we are the first to observe rapid growth of the serving infrastructure of a major content provider. We find that Google's serving infrastructure has almost doubled in the number of frontend IP addresses, has grown out to 62 countries, with serving sites deployed in 87 new ASes. Its recent growth strategy has been to move away from serving clients *off* its own backbone and towards serving from lower tiers in the AS hierarchy; the number of /24 prefixes served *off* Google's network nearly doubled during the expansion. Furthermore, these new serving sites, predictably, have narrow customer cones, serving only the customers of the AS the site is deployed in. Finally, we find that the expansion has noticeably shifted the distribution of geographic distances from the client to its nearest front-end server, and that this shift can also reduce the error in geolocating frontends using client locations alone, but not enough to obviate the need for CCG's filtering techniques.

2. BACKGROUND

CDNs and Serving Intrastructures. Adding even a few hundreds of milliseconds to a webpage load time can cost service providers users and business [30, 17], so providers seek to optimize their web serving infrastructure to deliver content quickly to clients. Whereas once a website might have been served from a single location to clients around the world, today's major services rely on much more complicated and distributed infrastructure. Providers replicate their services at serving sites around the world and try to serve a client from the closest one [15]. Content delivery networks (CDNs) initially sped delivery by caching static content and some forms of dynamic content within or near client networks.

Today, providers use this type of distributed infrastructure to speed the delivery of dynamic personalized content and responses to queries. To do so, providers direct clients to serving sites in or near the clients' networks. A client's TCP connection terminates at a frontend server in the serving site, but the frontend proxies the request back to one of the provider's large datacenters [25]. This arrangement has a number of potential advantages versus directing the client directly to the datacenter. For example, the client's latency to the frontend is less than the client's latency to the data center, allowing TCP to recover faster after loss, the primary cause of suboptimal performance. Moreover, the frontend can multiplex many clients into a high throughput connection to the datacenter.

In these types of serving infrastructures, different classes of serving sites may serve different clients. First, of course, the provider may still serve clients near a datacenter directly from that datacenter. Second, if a client network hosts a serving site, it will generally only allow its clients (or the clients of one of its customer networks) to use frontend servers in its site, not clients of its providers or peers.

DNS-based Redirection. Serving infrastructures use the Domain Name System (DNS) to direct clients to appropriate serving sites and frontend servers. When a client queries DNS to resolve a name associated with a service, the service returns an IP address for a frontend it believes is near the client. Traditionally, at resolution time, however, the service only knows the IP address of the client's resolver and not of the client itself, leading to two main complications. The resolver may be far from the clients it serves, and so the server closest to the resolver may not be a good choice for the client. Existing techniques can allow many services to discover which clients use a particular resolver [20], enabling services to direct a resolver based on the clients that use it. However, these techniques are of little benefit if the same resolver serves clients that are far from each other– there is no server that the service can return that will be a good choice for all clients who may have issued the request through the resolver.

To overcome this hurdle and provide quality DNS redirections for clients, a number of Internet providers and CDNs proposed EDNS-client-subnet [8]. EDNS is an IETF specification designed to overcome parameter size restrictions in standard DNS. EDNS-client-subnet is an experimental extension to EDNS that allows a client to embed a portion of its IP address in the request which will travel to an authoritative name server. By including the client IP prefix in the request, the extension allows a service to direct the client to the optimal server directly, without having to infer which client is behind a request from a recursive resolver.

3. GOAL AND APPROACH

Our goal is to understand content serving strategies for large IPv4-based serving infrastructures, especially that of Google. Serving strategies are defined by how

many serving sites and frontend servers a serving infrastructure has, where the serving sites are located geographically and topologically (i.e., within which ISP), and which clients access which serving sites. Furthermore, services continuously evolve serving strategies, so we are also interested in measuring the evolution of serving infrastructures. Of these, Google's serving infrastructure is arguably one of the most important, so we devote significant attention to this infrastructure.

To this end, we develop novel measurement methods to enumerate frontend servers, geolocate serving sites, and cluster frontend servers into serving sites. The challenge in devising these measurement methods is that serving infrastructures are large, distributed entities, with thousands of frontend servers at hundreds of serving sites spread across dozens of countries. A brute force approach to enumerating serving sites would require perspectives from a very large number of topological locations in the Internet, much larger than the geographic distribution provided by research measurement infrastructures like PlanetLab. Moreover, existing geolocation methods that rely on DNS naming or geolocation databases do not work well on these serving infrastructures where location-based DNS naming conventions are not consistently employed.

While our measurement methods use these research infrastructures for some of their steps, the key insight in the design of the methods is to *leverage mechanisms used by serving infrastructures to serve content.* Because we design them for serving infrastructures, these mechanisms can enumerate and geolocate serving sites more accurately than existing approaches, as we discuss below.

Our method to enumerate all frontend server IP addresses within the serving infrastructure uses the EDNS-client-subnet extension. As discussed in Section 2, Google (and some other serving infrastructures) use this extension to address the problem of geographically distributed clients using a resolver that prevents the serving infrastructure from optimally directing clients to frontends. We use this extension to enumerate frontend IP addresses of a serving infrastructure *from a single location*: this extension can emulate DNS requests coming from every active prefix in the IP address space, effectively providing a very large set of vantage points for enumerating frontend IP addresses.

To geolocate frontend servers and serving centers, we leverage another mechanism that serving infrastructures have long deployed. They have developed sophisticated mapping algorithms that maintain performance maps to clients with the goal of directing clients to the nearest available server. These algorithms have the property that clients that are directed to the server are likely to be topologically, and probably geographically, close to the server. We exploit this property to geolocate frontend servers: essentially, we approximate the location of a server by the geographical mean of client locations, a technique we call *client-centric geolocation* or CCG. We base our technique on this intuition, but we compensate for incorrect client locations and varying density of server deployments.

Finally, we leverage existing measurement infrastructure (PlanetLab) to cluster frontends into serving sites. We model the relative location of a frontend server as a vector of round-trip-times to many vantage points in the measurement infrastructure, then employ standard clustering algorithms in this high-dimensional space.

Using these measurement methods over a six month period, we are able to study Google's serving infrastructure and its evolution. Coincidentally, Google's deployments have doubled over this period, and we explore salient properties of this expansion: where (geographically or topologically) most of the expansion has taken place, and how it has impacted clients.

There are interesting aspects of Google's deployment that we currently lack means to measure. In particular, we do not know the query volume from different clients, and we do not know the latency from clients to servers (which may or may not correlate closely with the geographic distance that we measure). We have left exploration of these to future work. We do possess information about client *affinity* to frontend servers, and how this affinity evolves over time (this evolution is a function of improvements in mapping algorithms as well as infrastructure rollout): we have left a study of this to future work.

4. METHODOLOGY

In this section, we discuss the details of our measurement methods for enumerating frontends, geolocating them, and clustering them into serving sites.

4.1 Enumerating Frontends

Our first goal is to enumerate the IP addresses of all frontends within a serving infrastructure. We do not attempt to identify when multiple IP addresses belong to one computer, or when one address fronts for multiple physical computers. An IP addresses can front hardware from a small satellite proxy to a huge datacenter, so careful accounting of public IP addresses is not particularly meaningful.

Since most serving infrastructures use mapping algorithms and DNS redirection, one way to enumerate frontends is to issue DNS requests from multiple vantage points. Each request returns a frontend near the querying vantage point. The completeness of this approach is a function of the number of vantage points.

We emulate access to vantage points around the world using the proposed *client-subnet* DNS extension using the EDNS extension mechansim (we call this approach

EDNS-client-subnet). As of May 2013, EDNS-client-subnet is supported by Google, CacheFly, EdgeCast, ChinaCache and CDN 77. We use a patch to *dig*[1] that adds support for EDNS-client-subnet, allowing the query to specify the *client prefix*. In our measurements of Google, we issue the queries through Google Public DNS's public recursive nameservers, which passes them on to the service we are mapping. The serving infrastructure then return a set of frontends it believes are best suited for clients within the client prefix.

EDNS-client-subnet allows our our single measurement site to solicit the recommended serving infrastructure for all the Internet—we effectively get vantage points that are everywhere. We query using client prefixes drawn from 10 million routable /24 prefixes obtained RouteViews BGP. Queries against Google using this approach take about a day to enumerate.

4.2 Client-centric Geolocation

Current geolocation approaches are designed for generality, making few or no assumptions about the target. Unfortunately, this generality results in poor performance when geolocating serving infrastructure. For example, MaxMind's free database [21] places all Google frontends in Mountain View, the company's headquarters. General approaches such as CBG [10] work best when vantage points are near the target [14], but frontends in serving infrastructures are sometimes in remote locations, far from public geolocation vantage points. Techniques that use location hints in DNS names of frontends or routers near frontends can be incomplete [12].

Our approach combines elements of prior work, adding the observation that today's serving infrastructures use privileged data and advanced measurement techniques to try to direct clients to nearby frontends [31]. While we borrow many previously proposed techniques, our approach is unique and yields better results.

We base our geolocation technique on two main assumptions. First, a serving infrastructure tries to direct clients to a nearby frontend, although some clients may be directed to distant frontends, either through errors or a lack of deployment density. Second, geolocation databases have accurate locations for many clients, at least at country or city granularity, but also have poor granularity or erroneous locations for some clients.

Combining these two assumptions, our basic approach to geolocation, called *client-centric geolocation* (CCG), is to (1) enumerate the set of clients directed to a serving site, (2) query a geolocation database for the locations of those clients, and (3) assume the frontends are located geographically close to most of the clients.

To be accurate, CCG must overcome challenges inherent in each of these three steps of our basic approach:

1. We do not know how many requests different prefixes send to a serving infrastructure. If a particular prefix does not generate much traffic, the serving infrastructure may not have the measurements necessary to direct it to a nearby frontend, and so may direct it to a distant frontend.
2. Geolocation databases are known to have problems including erroneous locations for some clients and poor location granularity for other clients.
3. Some clients are not near the frontend that serve them, for a variety of reasons. For example, some frontends may serve only clients within certain networks, and some clients may have lower latency paths to frontends other than the nearest ones. In other cases, a serving infrastructure may direct clients to a distant frontend to balance load or may mistakenly believe that the frontend is near the client. Or, a serving infrastructure may not have any frontends near a particular client.

We now describe how CCG addresses these challenges.

Selecting client prefixes to geolocate a frontend. To enumerate frontends, CCG queries EDNS using all routable /24 prefixes. However, this approach may not be accurate for geolocating frontends, for the following reason. Although we do not know the details of how a serving infrastructure chooses which frontend to send a client to, we assume that it attempts to send a client to a nearby frontend and that the approach is more likely to be accurate for prefixes hosting clients who query the service a lot than for prefixes that do not query the service, such as IP addresses used for routers.

To identify which client prefixes can provide more accurate geolocation, CCG uses traceroutes and logs of users of a popular BitTorrent extension [7]. Ono issues traceroutes between connected pairs of users that provided an additional 102,064 prefixes with unlikely serving infrastructure mappings. From the user logs we obtain a list of 2 million client prefixes observed to participate in BitTorrent swarms with users. We assume that a serving infrastructure is likely to also observe requests from these prefixes.

Overcoming problems with geolocation databases. CCG uses two main approaches to overcome errors and limitations of geolocation databases. First, we exclude locations that are clearly wrong. Second, we combine a large set of client locations to locate each frontend and assume that the majority of clients have correct locations that will dominate the minority of clients with incorrect locations. To generate an initial set of client locations to use, CCG uses a BGP table snapshot from RouteViews [22] to find the set of prefixes currently announced, and breaks these routable prefixes up into 10 million /24 prefixes.[2] It then queries MaxMind's GeoLiteCity database to find locations for each /24 prefix.

[1] `http://wilmer.gaa.st/edns-client-subnet/`

[2] In Section 5.1, we verify that /24 is often the correct prefix length to use.

CCG prunes three types of prefix geolocations as untrustworthy. First, it excludes prefixes for which MaxMind indicates it has less than city-level accuracy. This heuristic excludes 1,966,081 of the 10 million prefixes (216,430 of the 2 million BitTorrent client prefixes). Second, it uses a dataset that provides coarse-grained measurement-based geolocations for every IP address to exclude prefixes that include addresses in multiple locations [11]. Third, it issues ping measurements from all PlanetLab locations to five responsive addresses per prefix, and excludes any prefixes for which the MaxMind location would force one of these ping measurements to violate the speed of light. Combined, these exclude 8,396 of the 10 million prefixes (2,336 of the 2 million BitTorrent client prefixes).

With these problematic locations removed, and with sets of prefixes likely to include clients, CCG assumes that both MaxMind and the serving infrastructure we are mapping likely have good geolocations for most of the remaining prefixes, and that the large number of accurate client geolocations should overwhelm any remaining incorrect locations.

Dealing with clients directed to distant frontends. Even after filtering bad geolocations, a client may be geographically distant from the frontend it is mapped to, for two reasons: the serving infrastructure may direct clients to distant frontends for load-balancing, and in some geographical regions, the serving infrastructure deployment may be sparse so that the frontend nearest to a client may still be geographically distant.

To prune these clients, CCG first uses speed-of-light constraints, as follows. It issues pings to the frontend from all PlanetLab nodes and use the speed of light to establish loose constraints on where the frontend could possibly be [10]. When geolocating the frontend, CCG excludes any clients outside of this region. This excludes 4 million out of 10 million prefixes (1.1 million out of 2 million BitTorrent client prefixes). It then estimates the preliminary location for the frontend as the weighted average of the locations of the remaining client prefixes, then refines this estimate by calculating the mean distance from the frontend to the remaining prefixes, and finds the standard deviation from the mean of the client-to-frontend distances. Our final filter excludes clients that are more than a standard deviation beyond the mean distance to the frontend, excluding 392,668 out of 10 million prefixes (214,097 out of 2 million BitTorrent client prefixes).

Putting it all together. In summary, CCG works as follows. It first lists the set of prefixes directed to a frontend, then filters out all prefixes except those observed to host BitTorrent clients. Then, it uses MaxMind to geolocate those remaining client prefixes, but excludes: prefixes without city-level MaxMind granularity; prefixes that include addresses in multiple locations; prefixes for which the MaxMind location is not in the feasible actual location based on speed-of-light measurements from PlanetLab and M-Lab; and prefixes outside the feasible location for the frontend. Its preliminary estimate for the frontend location is the geographic mean of the remaining clients that it serves. Calculating the distances from remaining clients to this preliminary location, CCG further exclude any clients more than a standard deviation beyond the mean distance in order to refine our location estimate. Finally, it locates the frontend as being at the geographic mean of the remaining clients that it serves.

4.3 Clustering frontends

As we discuss later, CCG is accurate to within 10s of kilometers. In large metro areas, some serving infrastructures may have multiple serving sites, so we develop a methodology to determine physically distinct serving sites. We cluster by embedding each frontend in a higher dimensional metric space, then clustering the frontend in that metric space. Such an approach has been proposed elsewhere [19, 34, 24] and our approach differs from prior work in using better clustering techniques and more carefully filtering outliers.

In our technique, we map each frontend to a point in high dimensional space, where the coordinates are RTTs from *landmarks* (in our case, 250 PlanetLab nodes at different geographical sites). The intuition underlying our approach is that two frontends at the same physical location should have a small distance in the high-dimensional space.

Each coordinate is the smallest but one RTT of 8 consecutive pings, and we use the Manhattan distance between two points for clustering. In computing this Manhattan distance, we (a) omit coordinates for which we received fewer than 6 responses to pings and (b) omit the highest 20% of coordinate distances to account for outliers caused by routing failures, or by RTT measurements inflated by congestion. Finally, we normalize this Manhattan distance.

The final step is to cluster frontends by their pairwise normalized Manhattan distance. We use the OPTICS algorithm [3] for this. OPTICS is designed for spatial data, and, instead of explicitly clustering points, it outputs an ordering of the points that captures the density of points in the dataset. As such, OPTICS is appropriate for spatial data where there may be no a priori information about either the number of clusters or their size, as is the case for our setting. In the output ordering, each point is annotated with a *reachability distance*: when successive points have significantly different reachability distances, that is usually an indication of a cluster boundary. As we show in Section 5 this technique, which dynamically determines cluster boundaries, is essential to achieving good accuracy.

	IPs	/24s	ASes	Countries
Open resolver	5182	275	131	59
EDNS-client-subnet	7040	365	169	60
Benefit	+36%	+33%	+29%	+2%

Table 1: Comparison of Google frontends found by EDNS and open resolver.

5. VALIDATION

In this section, we validate frontend enumeration, geolocation, and clustering.

5.1 Coverage of Frontend Enumeration

Using EDNS-client-subnet can improve coverage over previous methods that have relied on using fewer vantage points. We first quantify the coverage benefits of EDNS-client-subnet. We then explore the sensitivity of our results to the choice of prefix length for EDNS-client-subnet, since this choice can also affect front-end enumeration.

Open Resolver vs EDNS-client-subnet Coverage. An existing technique to enumerate frontends for a serving infrastructure is to issue DNS queries to the infrastructure from a range of vantage points. Following previous work [12], we do so using open recursive DNS (rDNS) resolvers. We use a list of about 200,000 open resolvers[3]; each resolver is effectively a distinct vantage point. These resolvers are in 217 counties, 14,538 ASes, and 118,527 unique /24 prefixes. Enumeration of Google via rDNS takes about 40 minutes. This dataset forms our comparison point to evaluate the coverage of the EDNS-client-subnet approach we take in this paper.

Table 1 shows the added benefit over rDNS of enumerating Google frontends using EDNS-client-subnet. Our approach uncovers at least 29% more Google frontend IP addresses, prefixes, and ASes than were visible using previous approaches. By allowing us to query Google on behalf of every client prefix, we obtain a view from locations that lack open recursive resolvers. In Section 6.1, we demonstrate the benefit over time as Google evolves, and in Section 8 we describe how we might be able to use our Google results to calibrate how much we would miss using rDNS to enumerate a (possibly much larger or smaller than Google) serving infrastructure that does not support EDNS-client-subnet.

EDNS-client-subnet Prefix Length. The choice of prefix length for EDNS-client-subnet can affect enumeration completeness. Prefix lengths smaller than /24 in BGP announcements are too coarse for enumeration. We find cases of neighboring /24s within shorter BGP announcement prefixes that are directed to different serving infrastructure. For instance we observed an ISP announcing a /18 with one of its /24 prefix getting directed to Singapore while its neighboring prefix is directed to Hong Kong.

Our evaluations query using one IP address in each /24 block. If serving infrastructures are doing redirections at finer granularity, we might not observe some frontend IP addresses or serving sites. The reply to the EDNS-client-subnet query returns the prefix length covering the response. Thus, if a query for an IP address in a /24 block returns a prefix length of, say /26, it means that the corresponding redirection holds for all IP addresses in the /26 covering the query address, not the /24. For almost 75% of our /24 queries, the other responses were for a /24 subnet, likely because it is the longest globally routable prefix. For most of the rest, we saw a /32 prefix length in the response, indicating that Google's serving infrastructure might be doing very fine-grained redirection. For each such /24 subnet (about 1/2 million subnets), we queried a 6-8 other IP addresses within that prefix, and we discovered only 3 additional IP addresses. Thus, we believe our choice of /24 minimally affects completeness, but we plan to understand the reasons for these fine-grain redirections in future work.

5.2 Accuracy of Client-Centric Geolocation

Client-centric geolocation using EDNS-client-subnet shows substantial improvement over traditional ping based techniques [10], undns [29], and geolocation databases [21].

Dataset. To validate our approach, we use the subset of Google frontends with hostnames that contain airport codes hinting at their locations. Although the airport location is not a precise location, we believe that it is reasonable to assume that the actual frontend is within a few 10s of kilometers. Using approximately 550 frontends with airport codes, we measure the error of our technique as the distance between our estimated location and the airport location.

Accuracy. Figure 1 shows the distribution of error for CCG, as well as for three traditional techniques. We compare to constraint-based geolocation (CBG), which uses latency-based constraints from a range of vantage points [10], a technique that issues traceroutes to frontends and locates the frontends based on geographic hints in names of nearby routers [12], and the MaxMind GeoLite Free database [21]. We offer substantial improvement over existing approaches. For example, the worst case error for CCG is 409km, whereas CBG, the traceroute-based technique, and MaxMind have errors of over 500km for 17%, 24%, and 94% of frontends, respectively. CBG performs well when vantage points are close to the frontend [14], but it incurs large errors for the half of the frontends in more remote regions. The traceroute-based technique is unable to provide any location for 20% of the frontends because there were no

[3]Used with permission from Duane Wessels, Packet Pushers Inc.

hops with geographic hints in their hostnames near to the frontend. The MaxMind database performs poorly because it places most frontends belonging to Google in Mountain View, CA.

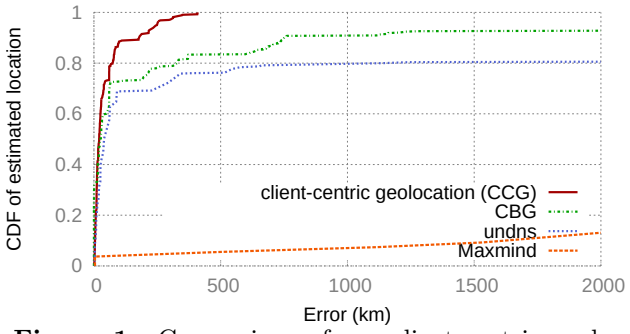

Figure 1: Comparison of our client-centric geolocation against traditional techniques, using Google frontends with known locations as ground truth.

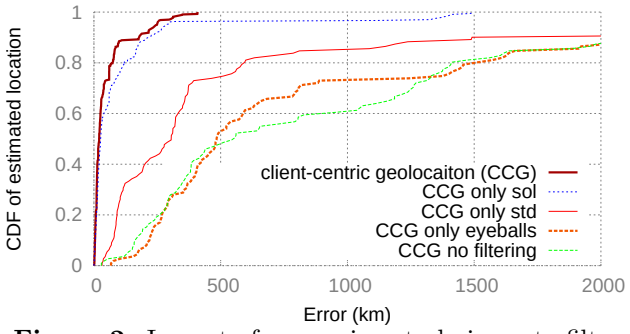

Figure 2: Impact of our various techniques to filter client locations when performing client-centric geolocation on Google frontends with known locations.

Importance of Filtering. Figure 2 demonstrates the need for the filters we apply in CCG. The *CCG no filtering* line shows our basic technique without any filters, yielding a median error of 556km. Only considering client eyeball prefixes we observed in the BitTorrent dataset reduces the median error to 484km and increases the percentage of frontends located with error less than 1000km from 61% to 74%. Applying our standard deviation filtering improves the median to 305km and error less than 1000km to 86%. When using speed-of-light constraints measured from PlanetLab and MLab to exclude client locations outside the feasible location for a frontend and to exclude clients with infeasible MaxMind locations, we obtain a median error of 26km, and only 10% of frontend geolocations have an error greater than 1000km. However, we obtain our best results by simultaneously applying all three filters.

Case Studies of Poor Geolocation. CCG's accuracy depends upon its ability to draw tight speed-of-light constraints, which in turn depends (in our current implementation), on Planetlab and M-Lab deployment density. We found one instance where sparse vantage point deployments affected CCG's accuracy. In this instance, we observe a set of frontends in Stockholm, Sweden, with the *arn* airport code, serving a large group of client locations throughout Northern Europe. However, our technique locates the frontends as being 409km southeast of Stockholm, pulled down by the large number of clients in Oslo, Copenhagen and northern Germany. Our speed of light filtering usually effectively eliminates clients far from the actual frontend. In this case, we would expect Planetlab sites in Sweden to filter out clients in Norway, Denmark and Germany. However, these sites measure latencies to the Google frontends in the 24ms range, yielding a feasible radius of 2400km. This loose constraint results in poor geolocation for this set of frontends.

It is well-known that Google has a large datacenter in The Dalles, Oregon, and our map (Fig. 7) does not show any sites in Oregon. In fact, we place this site 240km north, just south of Seattle, Washington. A disadvantage of our geolocation technique is that large data centers are often hosted in remote locations, and our technique will pull them towards large population centers that they serve. In this way, the estimated location ends up giving a sort of "logical" serving center of the server, which is not always the geographic location.

5.3 Accuracy of Frontend Clustering

To validate the accuracy of our clustering method, we run clustering on three groups of nodes for which we have ground truth: 72 PlanetLab servers from 23 different sites around world; 27 servers from 6 sites all in California, USA, some of which are very close (within 10 miles) to each other, within 10 miles; and finally, 75 Google IP addresses that have 9 different airport codes in their reverse DNS names. These three sets are of different size and geographic scope, and the last set is a subset of our target so we expect it to be most representative.

The metric we use for the accuracy of clustering is the *Rand Index* [26]. The index is measured as the ratio of the sum of true positives and negatives to the ratio of the sum of these quantities *and* false positives and negatives. A Rand index equal to 1 means there are *no* false positives or false negatives.

Table 2 shows the Rand index for the 3 node sets for which we have ground truth. We see that in each case, the Rand index is upwards of 97%. This accuracy arises from two components of the design of our clustering method: eliminating outliers which result in more accurate distance measures, and dynamically selecting the cluster boundary using our OPTICS algorithm.

Our method does have a small number of false pos-

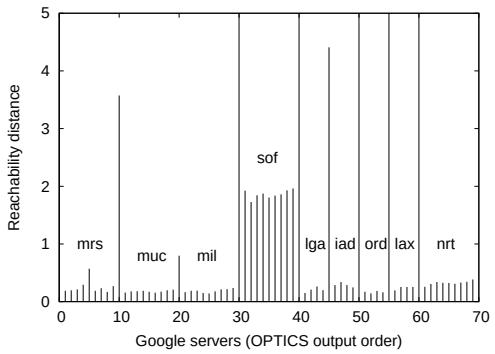

Figure 3: Distance plot of Google servers with airport codes. Servers in the same cluster have low reachability distance to each other thus are output in sequence as neighbors. Cluster boundaries are demarcated by large impulses in the reachability plot.

Experiment	Rand Index
PlanetLab	0.99
CA	0.97
Google	0.99 or 1

Table 2: Rand index for our nodesets. Our clustering algorithm achieves over 97% across all nodesets, indicating very few false positives or negatives.

itives and false negatives. In the California nodeset, the method fails to set apart some USC/ISI nodes from nodes on the USC campus, and in the Planet lab nodeset, some clusters have low reachability distance that confuses our boundary detection method. The Google nodeset reveals one false negative which we actually believe to be correct: the algorithm correctly identifies two distinct serving sites in `mrs`, as discussed below.

To better understand the performance of our method, Figure 3 shows the output of the OPTICS algorithm on the Google nodeset. The x-axis in this figure represents the ordered output of the OPTICS algorithm, and the y-axis the reachability distance associated with each node. Impulses in the reachability distance depict cluster boundaries, and we have verified that the nodes within the cluster all belong to the same airport code. In fact, as the figure shows, the algorithm is correctly able to identify all 9 Google sites. More interesting, it shows that, within a single airport code `mrs`, there are likely two physically distinct serving sites. We believe this to be correct, from an analysis of the DNS names associated with those front-ends: all frontends in one serving site have a prefix `mrs02s04`, and all frontends in the other serving site have a prefix `mrs02s05`.

Finally, Figure 4 shows the OPTICS output when using reverse-TTL (as proposed in [19]) instead of RTT for the metric embedding. This uses a slightly different set

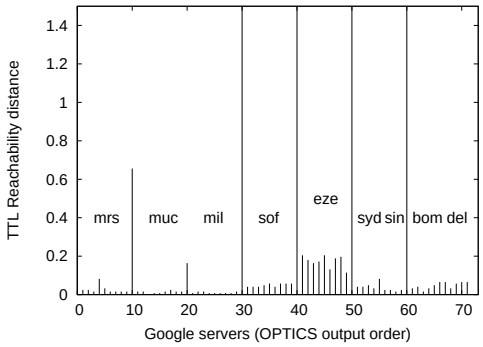

Figure 4: The output of the OPTICS clustering algorithm when reverse-TTL is used for the metric embedding. When using this metric, the clustering algorithm cannot distinguish serving sites at Bombay (`bom`) and Delhi (`del`) in India, while RTT-based clustering can.

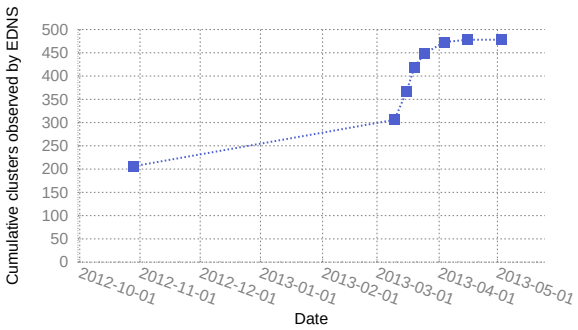

Figure 6: Growth in the number of points of presence hosting Google serving infrastructure over time.

of Google servers than in our evaluation above: this set was chosen to highlight the performance of reverse-TTL based clustering. For this set of nodes, reverse-TTL based embedding performs reasonably well but results in the OPTICS algorithm being unable to distinguish between serving sites in `bom` and `del`. RTT-based clustering is able to differentiate these serving sites (not shown). Moreover, although reverse-TTL suggests the possibility of two sites in `mrs`, it mis-identifies which servers belong to which of these sites (based on reverse DNS names).

6. MAPPING GOOGLE'S EXPANSION

We present a longitudinal study of Google's serving infrastructure. Our initial dataset is from late October to early November of 2012 and our second dataset covers March and April of 2013. We are able to capture a substantial expansion of Google infrastructure.

6.1 Growth over time

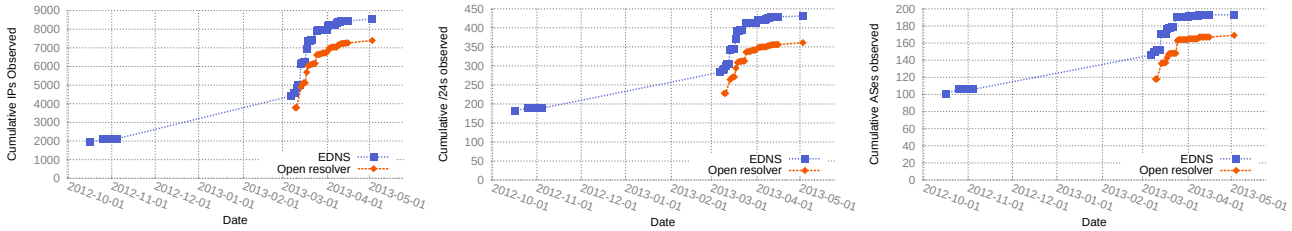

Figure 5: Growth in the number of IP addresses (a), /24 prefixes (b), and ASes/countries (c) observed to be serving Google's homepage over time. During our study, Google expanded rapidly at each of these granularities.

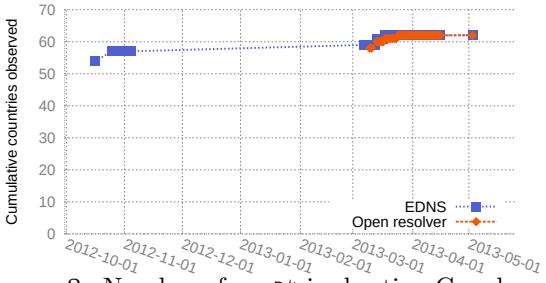

Figure 8: Number of countries hosting Google serving infrastructure over time.

	November 2012		May 2013			
	ASes	Clients	ASes		Clients	
Google	2	9856K	2	(+0%)	9658K	(-2%)
Tier 1	2	481	2	(+0%)	201	(-58%)
Large	30	111K	46	(+53%)	237K	(+114%)
Small	35	37K	64	(+83%)	63K	(+71%)
Tiny	23	31K	41	(+78%)	57K	(+84%)
Stub	13	21K	36	(+177%)	38K	(+81%)

Table 3: Classification of ASes hosting Google serving infrastructure at the beginning and end of our study. We count both by the number of distinct ASes and by the number of client /24 prefixes served. Although Google still directs 96% of the 10 million prefixes to servers within its own network, it is evolving towards serving fewer clients from its own network and more clients from smaller ASes around the world.

For each snapshot that we capture, we use EDNS-client-subnet to enumerate all IP addresses returned for `www.google.com`. Figure 5(a) depicts the number of server IP addresses seen in these snapshots over time.[4] The graph shows slow growth in the cumulative number of Google IP addresses observed between November 2012 and March 2013, then a major spike in mid-March in which we saw approximately 3000 new serving IP addresses come online. By the end of our study, the number of serving IP addresses tripled. Figure 5(b) shows this same trend in the growth of the number of /24s seen to serve Google's homepage. In Figure 5(c), we see 82% growth in the number of ASes originating these prefixes, indicating that this large growth is not just Google adding new capacity to existing serving locations. Figure 6 shows the growth in the number of distinct serving sites within those ASes.

Figure 7 shows the geographic locations of Google's serving infrastructure at the beginning of our measurements and in our most recent snapshot. We observe two types of expansion. First, we see new serving locations in remote regions of countries that already hosted servers, such as Australia and Brazil. Second, we observe Google turning up serving infrastructure in countries that previously did not appear to serve Google's homepage, such as Vietnam and Thailand. Of new frontend IP addresses that appeared during the course of our study, 92% are in ASes other than Google. Of those addresses, only 13% are in the United States or Europe, places that are well-served directly from Google's network. Outside these regions, 45% are in Asia, 23% in North America (outside the US), 20% are in South America, and 8% are in Africa. Figure 8 depicts this growth in the number of countries hosting serving infrastructure, from 53 or 56 at the beginning of our study to 62 in recent measurements. We intend to continue to run these measurements indefinitely to continue to map this growth.

6.2 Characterizing the Expansion

To better understand the nature of Google's expansion, we examine the types of networks where the expansion is occurring and how many clients they serve. Table 3 classifies the number of ASes of various classes in which we observe serving infrastructure, both at the beginning and at the end of our study. It also depicts the number of /24 client prefixes (of 10 million total) served by infrastructure in each class of AS. We use AS classifications from the June 28, 2012 dataset from UCLA's Internet Topology Collection [33],[5] except that we only classify as stubs ASes with 0 costumers, and we introduce a Tiny ISP class for ASes with 1-4 customers.

As seen in the table, the rapid growth in ASes that

[4]It is not necessarily the case that each IP address maps to a distinct frontend.

[5]UCLA's data processing has been broken since 2012, but we do not expect the AS topology to change rapidly.

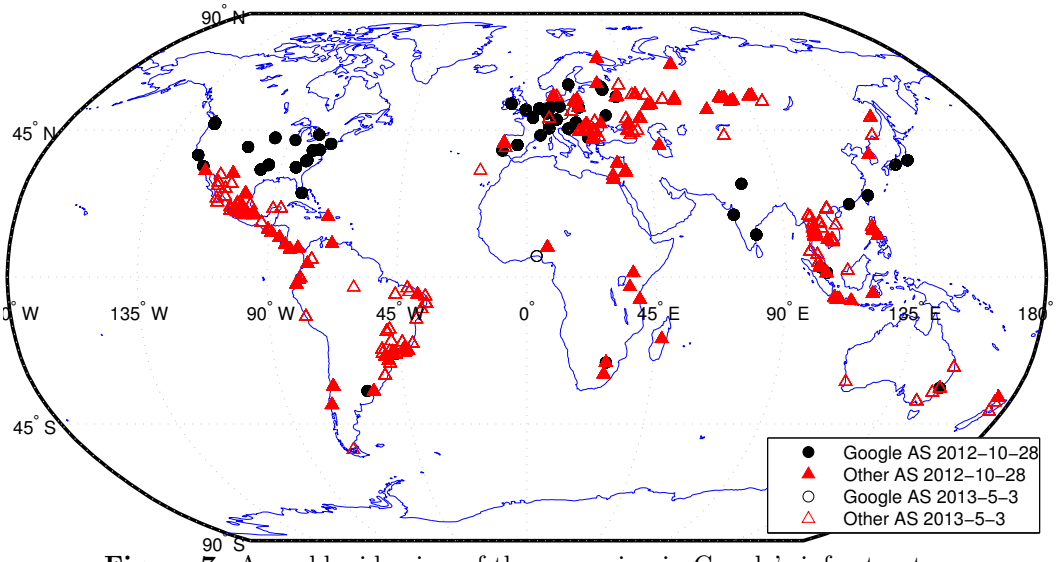

Figure 7: A world wide view of the expansion in Google's infrastructure.

host infrastructure has mainly been occurring lower in the AS hierarchy. Although Google still directs the vast majority of client prefixes to servers in its own ASes, it has begun directing an additional 2% of them to servers off its network, representing a 98% increase in the number served from off the network. By installing servers inside client ISPs, Google allows clients in these ISPs to terminate their TCP connections locally (likely at a satellite server that proxies requests to a datacenter [25], as it is extremely unlikely that Google has sufficient computation in these locations to provide its services). We perform reverse DNS lookups on the IP addresses of all frontends we located outside of Google's network. More than one third of them have hostnames that include either *ggc* or *google.cache*. These results suggest that Google is reusing infrastructure from the Google Global Cache (GGC), Google's content distribution network built primarily to cache YouTube videos near users.[6]

Figure 9 depicts a slightly different view of the Google expansion. It charts the cumulative distribution of the number of serving sites by ISP type. Almost half of the ISPs, by any type, host only one serving site. Generally speaking, smaller ISPs host fewer serving sites than larger ISPs, with some large ISPs hosting up to 10 different sites. The one exception is a Tiny ISP in Mexico hosting 20 serving sites consisting of hundreds of frontend IPs. We are currently examining this outlier in detail.

Whereas Google would be willing to serve any client from a server located within the Google network, an ISP hosting a server would likely only serve its own customers. Serving its provider's other customers, for example, would require the ISP to pay its provider for the service! We check this intuition by comparing the location in the AS hierarchy of clients and the servers to which Google directs them. Of clients directed to servers outside of Google's network, 90% are located within the server's AS's customer cone (the AS itself, its customers, their customers, and so on) [18]. Since correctly inferring AS business relationship is known to be a hard problem [9], it is unclear whether the remaining 10% of clients are actually served by ISPs of which they are not customers, or (perhaps more likely) whether they represent limitations of the analysis. In fact, given that 60% of the non-customer cases stem from just 4 serving ASes, a small number of incorrect relationship or IP-to-AS inferences could explain the counter-intuitive observations.

Google's expansion of infrastructure implies that, over time, many clients should be directed to servers that are closer to them than where Google directed them at the

Figure 9: CDF of number of clusters in different types of ISP

[6]GGC documentation mentions that the servers may be used to proxy Google Search and other services.

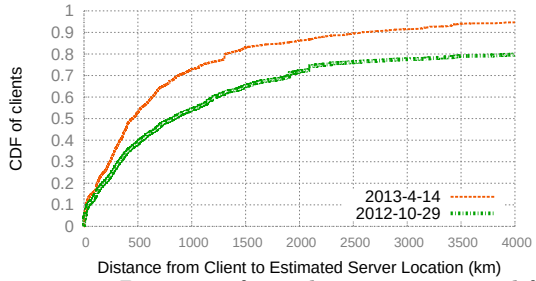

Figure 10: Distances from clients to estimated frontend locations to which Google directs them.

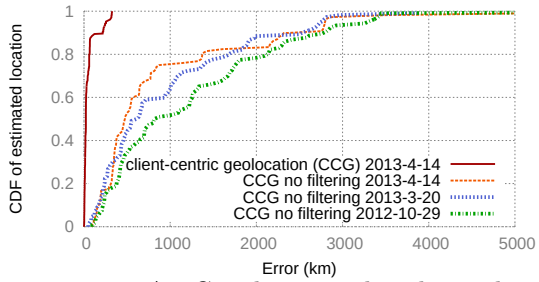

Figure 11: As Google expands, clients become closer to their servers, improving accuracy of filterless client-based geolocation.

beginning of the study. Figure 10 shows the distribution of the distance from a client to our estimate of the location of the server serving it. We restrict the clients to those in our BitTorrent eyeball dataset and geolocate all client locations using MaxMind. Some of the very large distances shown in both curves could be accuracy limitations of the MaxMind GeoLite Free database, especially in regions outside of the United States. Overall, results show that in mid-April 2013, many clients are substantially closer to the set of servers they are directed to than in October of 2012. For example, the fraction of client prefixes within 500km of their servers increases from 39% to 53%, and the fraction within 1000km increases from 54% to 73%. Because many of the newer frontends seem to be satellites that likely proxy traffic back to datacenters, it is hard to know the impact that decreasing the distance from client to frontend will have on application performance [25].

6.3 Impact on Geolocation Accuracy

A side-effect of Google directing more clients to frontends closer to them is that our geolocation technique should become more accurate over time, since we base it on the assumption that frontends are near their clients. To verify that assumption, we apply our basic geolocation approach–without any of our filters that increase accuracy–to the datasets from three points in time. We chose dates to coincide with the large jumps in Google servers that we observe in Figure 5. Using the airport code-based ground truth dataset from Section 5.2, Figure 11 shows the distribution of error in geolocation using these three datasets and, for comparison, the most recent dataset using all our filters. We can see that there is steady reduction in error over time, with median error decreasing from 817km in October 2012, to 610km in March 2013, and 475km in April 2013. However, our filters still provide substantial benefit, yielding a median error of only 22km.

7. RELATED WORK

Closest to our work is prior work on mapping CDN infrastructures [12, 2, 32, 1]. Huang et al. [12] map two popular content delivery networks, Akamai and Limelight, by enumerating their frontends using a quarter of a million open rDNS resolvers. They geolocate and cluster frontends using a geolocation database as well as using the location of penultimate hop of traceroutes to frontends. Ager et al. [2] chart web hosting structures as a whole. They start from probing several sets of domain names from dozens of vantage points to collect service IP addresses. They rely entirely on MaxMind [21] for geolocation, and use feature-based clustering where the goal of clustering is to separate frontends belonging to different hosting infrastructures. Torres et al. [32] use a small number of vantage points in the US and Europe and constraint-based geolocation to approximately geolocate serving sites in the YouTube CDN, with the aim of understanding video server selection strategies. Finally, Adhikari et al. [1] use open resolvers to enumerate YouTube servers and geolocation databases to geolocate them, with the aim of reverse-engineering the caching hierarchy and logical organization of YouTube infrastructure using DNS namespaces.

In contrast to these pieces of work, our enumeration effectively uses many more vantage points, our geolocation technique leverages client locations for accuracy instead of relying on geolocation databases, and our clustering technique relies on a metric embedding in high-dimensional space to differentiate between nearby sites.

Several other pieces of work are tangentially related to ours. Mao et al. [20] quantifies the proximity of clients to their local DNS resolvers and finds that clients in different geographic locations may use the same resolver. The EDNS-client-subnet extension we use was designed to permit serving infrastructures to more accurately direct clients to serving sites in these cases. Other work [31, 7] has exploited the observation that two clients directed to the same or nearby frontends are likely to be geographically close. Our work uses this observation to geolocate frontends. Otto et al. [23] examine the end to end impact that different DNS ser-

vices have on CDN performance. It is the first work to study the potential of the EDNS-client-subnet to address the client CDN mapping problem, but does not attempt to map Google's expansion, as we do.

Finally, several strands of research have explored complementary problems associated with serving infrastructures, ranging from characterizing and diagnosing latency of CDNs [35, 15] as well as cloud providers [16] and search services [6], to geolocating ASs using client locations [27], verifying data replication strategies for cloud providers [4], analyzing content usage in large CDNs [5].

8. USING OUR MAPPING

In addition to our evaluation of Google's serving infrastructure so far, our mapping is useful to the research community, for what it says about clients, and for what it can predict about other serving infrastructure.

The Need for Longitudinal Research Data. Our results show the limitations of one-off measurement studies—a snapshot of Google's serving infrastructure in October would have missed the rapid growth of their infrastructure and potentially misrepresented their strategy. We believe the research community needs long-term measurements, and we intend to refresh our maps regularly. We will make our ongoing data available to the research community, and we plan to expand coverage from Google to include other providers' serving infrastructures.

Sharing the Wealth: From Our Data to Related Data. Our mapping techniques assume the target sharing infrastructure is pervasive and carefully and correctly engineered. We assume that (a) Google directs most clients to nearby frontends; (b) Google's redirection is carefully engineered for "eyeball" prefixes that host end-users; and (c) Google will only direct a client to a satellite frontend if the client is a customer of the frontend's AS. Google has economic incentives to ensure these assumptions. In practice, these assumptions are generally true but not always, and our design and evaluation has carefully dealt with exceptions (such as clients occasionally being directed to distant frontends.

If we accept these assumptions, our maps allow us to exploit Google's understanding of network topology and user placement to improve other datasets. Prior work has used Akamai to chose detour routes [31]; we believe our mapping can improve geolocation, peer selection, and AS classification.

Geolocation is a much studied problem [11, 10, 14], and availability of ground truth can greatly improve results. With clients accessing Google from mobile devices and computers around the world, Google has access to ample data and measurement opportunity to gather very accurate client locations. An interesting future direction is to infer prefix location from our EDNS-client-subnet observations, and use that coarse data to re-evaluate prefixes that existing datasets (such as MaxMind) place in very different locations. The end result would be either higher accuracy geolocation or, at least, identification of prefixes with uncertain locations.

Researchers designed a BitTorrent plugin that would direct a client to peer with other users the plugin deemed to be nearby, because the potential peer received similar CDN redirections as the client's [7]. However, with the existing plugin, the client can only assess similarity of other users of the plugin who send their CDN frontend mappings. Just as we used EDNS-client-subnet to obtain mappings from arbitrary prefixes around the world, we could design a modified version of the plugin that would allow a client to assess the nearness of an arbitrary potential peer, regardless of whether the peer uses the plugin or not. By removing this barrier, the modified plugin would be much more widely applicable, and could enhance the adoption of such plugins.

Finally, in Section 6.2, we showed that 90% of prefixes served in ASes other than Google are within the customer cone of their serving AS. The remaining 10% of prefixes likely represent problems with either our IP-to-AS mapping [13] or with the customer cone dataset we used [18]. From talking to the researchers behind that work and sharing our results with them, it may be necessary to move to prefix-level cones, to accommodate the complex relationships between ASes in the Internet. The client-to-frontend data we generate could help resolve ambiguities in AS relationships and lead to better inference in the future.

Calibrating Other Measurements. Our studies of Google combine observations using EDNS-client-subnet and open recursive resolvers. Not all providers support EDNS-client-subnet, however.

In Section 5.1, we demonstrated that even using hundreds of thousands of open recursive DNS resolvers would miss discovering much of Google's infrastructure that we uncover using EDNS-client-subnet. We next consider how we can use our results from Google to project results for other providers that support only open resolvers.

To explore the feasibility of this projection, Figure 12 depicts the number of Google IP addresses discovered as we issue additional measurements. We select one open recursive resolver from each /24 in which we know of at least one resolver (there are 110,000 such prefixes). Then, we select one of these /24s at a time and resolve `www.google.com` from the open resolver in the prefix and via an EDNS-client-subnet query for that prefix. The figure depicts the growth in the number of Google frontend IP addresses discovered by the two approaches (min, mean, and max over 1000 trials). As seen in the figure, using resolvers in a set of prefixes yields very

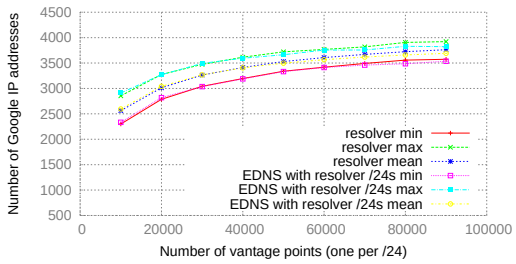

Figure 12: The relation between number of Google IP addresses discovered and the number of vantage points. Using one open resolver per /24 block and one EDNS query per /24 block.

similar numbers of frontend IPs to issuing EDNS-client-subnet queries for that same set of prefixes, so that the benefit of EDNS-client-subnet is primarily that we can issue queries for many more prefixes than we have access to resolvers in.

This suggests that we may be able to extrapolate these growth curves to understand the impact of having more resolvers. To test this theory, we fit power law curves to the open resolver lines ($R = 0.97$ in all cases). We project that access to resolvers in all 10M routable /24 prefixes, predicting discovery of 6990–8687 IP addresses of Google frontend servers. Using EDNS-client-subnet queries for these 10M prefixes, we found 8563 IP addresses, within the range, suggesting that the extrapolation approach may be reasonable. In the future, we plan to apply it to predict the size of Akamai and other infrastructures that do not yet support EDNS-client-subnet.

9. CONCLUSIONS

As the role of interactive web applications continues to grow in our lives, and the mobile web penetrates remote regions of the world more than wired networks ever had, the Internet needs to deliver fast performance to everyone, everywhere, at all times. To serve clients around the work quickly, service providers deploy globally distributed serving infrastructure, and we must understand these infrastructures to understand how providers deliver content today. Towards that goal, we developed approaches specific to mapping these serving infrastructures. By basing our techniques around how providers architect their infrastructures and guarding our techniques against noisy data, we are able to accurately map the geographically-distributed serving sites.

We apply our techniques to mapping Google's serving infrastructure and track its rapid expansion over the period of our measurement study. During that time, the number of serving sites doubled, and we see Google deploying satellite frontends around the world, in many cases distant from any known Google datacenters. By continuing to map Google's and others' serving infrastructures, we will watch the evolution of these key enablers of today's Internet, and we expect the accurate maps to enable future work by us and others to understand and improve content delivery on a global scale.

10. REFERENCES

[1] Vijay Kumar Adhikari, Sourabh Jain, Yingying Chen, and Zhi-Li Zhang. Vivisecting YouTube: An active measurement study. In *INFOCOM, 2012 Proceedings IEEE*, pages 2521–2525. IEEE, 2012.

[2] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. Web content cartography. In *Proc. of ACM Intermet Measurement Conference*, pages 585–597, Berlin, Germany, November 2011. ACM.

[3] Mihael Ankerst, Markus M. Breunig, Hans-peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. In *ACM SIGMOD international conference on Management of data*, pages 49–60, Philadelphia, PA, USA, June 1999. ACM.

[4] Karyn Benson, Rafael Dowsley, and Hovav Shacham. Do you know where your cloud files are? In *Proc. of Cloud Computing Security Workshop*, pages 73–82, Chicago, Illinois, USA, October 2011. ACM.

[5] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.

[6] Yingying Chen, Sourabh Jain, Vijay Kumar Adhikari, and Zhi-Li Zhang. Characterizing roles of front-end servers in end-to-end performance of dynamic content distribution. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 559–568. ACM, 2011.

[7] David Choffnes and Fabian E. Bustamante. Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *SIGCOMM*, pages 363–374, Seattle, WA, USA, October 2008.

[8] C. Contavalli, W. van der Gaast, S. Leach, and E. Lewis. Client subnet in dns requests, April 2012. Work in progress (Internet draft draft-vandergaast-edns-client-subnet-01).

[9] Xenofontas Dimitropoulos, Dmitri Krioukov,

Marina Fomenkov, Bradley Huffaker, Young Hyun, k. c. claffy, and George Riley. AS relationships: Inference and validation. *ACM Computer Communication Review*, 37(1):29–40, January 2007.

[10] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-based geolocation of Internet hosts. *IEEE/ACM TON*, 14(6):1219–1232, December 2006.

[11] Zi Hu and John Heidemann. Towards geolocation of millions of IP addresses. In *Proc. of ACM Intermet Measurement Conference*, pages 123–130, Boston, MA, USA, 2012. ACM.

[12] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Measuring and evaluating large-scale CDNs. Technical Report MSR-TR-2008-106, Microsoft Research, October 2008.

[13] iPlane. `http://iplane.cs.washington.edu`.

[14] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards IP geolocation using delay and topology measurements. In *IMC*, pages 71–84, 2006.

[15] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving beyond end-to-end path information to optimize CDN performance. In *IMC*, 2009.

[16] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp: comparing public cloud providers. In *Proc. of ACM Intermet Measurement Conference*, pages 1–14. ACM, 2010.

[17] Greg Linden. Make data useful. `http://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-28.ppt`, 2006.

[18] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, and k claffy. AS Relationships, Customer Cones, and Validation. submitted to IMC 2013.

[19] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, 2006.

[20] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J Wang. A Precise and Efficient Evaluation of the Proximity Between Web Clients and their Local DNS Servers. *USENIX Annual Technical Conference*, pages 229–242, 2002.

[21] MaxMind. `http://www.maxmind.com/app/ip-location/`.

[22] David Meyer. RouteViews. `http://www.routeviews.org`.

[23] John S. Otto, Mario A. Sánchez, John P. Rula, and Fabián E Bustamante. Content delivery and the natural evolution of DNS. In *Proc. of ACM Intermet Measurement Conference*, Boston, Massachusetts, USA, November 2012. ACM.

[24] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *Proc. ofACM SIGCOMM*, pages 173–185, San Diego, California, USA, August 2001. ACM.

[25] Abhinav Pathak, Y. Angela Wang, Cheng Huang, Albert Greenberg, Y. Charlie Hu, Randy Kern, Jin Li, and Keith W. Ross. Measuring and evaluating TCP splitting for cloud services. In *PAM*, 2010.

[26] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[27] Amir H Rasti, Nazanin Magharei, Reza Rejaie, and Walter Willinger. Eyeball ASes: from geography to connectivity. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 192–198. ACM, 2010.

[28] Steve Souders. High-performance web sites. *Communications of the ACM*, 51(12):36–41, December 2008.

[29] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[30] Stoyan Stefanov. Yslow 2.0. In *CSDN SD2C*, 2008.

[31] Ao-Jan Su, David R. Choffnes Aleksandar Kuzmanovic, and Fabi'an E. Bustamante. Drafting behind Akamai (Travelocity-based detouring). In *Proc. ofACM SIGCOMM*, pages 3–14, Pisa, Italy, September 2006. ACM.

[32] Ruben Torres, Alessandro Finamore, Jin Ryong Kim, Marco Mellia, Maurizio M Munafo, and Sanjay Rao. Dissecting video server selection strategies in the YouTube CDN. In *31st International Conference on Distributed Computing Systems (ICDCS)*, pages 248–257. IEEE, 2011.

[33] UCLA Internet topology collection. `http://irl.cs.ucla.edu/topology/`.

[34] Qiang Xu and Jaspal Subhlok. Automatic clustering of grid nodes. In *Proc. of6th IEEE International Workshop on Grid Computing*. IEEE, November 2005.

[35] Yaping Zhu, Benjamin Helsley, Jennifer Rexford, Aspi Siganporia, and Sridhar Srinivasan. LatLong: Diagnosing wide-area latency changes for CDNs. *IEEE Transactions on Network and*

Service Management, 9(1), September 2012.