Mapping the Expansion of Google's Serving Infrastructure[*]

Matt Calder
University of Southern California

Xun Fan
USC/ISI

Zi Hu
USC/ISI

Ethan Katz-Bassett
University of Southern California

John Heidemann
USC/ISI

Ramesh Govindan
University of Southern California

ABSTRACT

Modern content-distribution networks both provide bulk content and act as "serving infrastructure" for web services in order to reduce user-perceived latency. 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 IP addresses of servers in these infrastructures, find their geographic location, and identify the association between clients and clusters of 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 DNS extension to measure which clients a service maps to which of its serving sites. 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 accuracy relative to existing approaches. We also cluster server IP addresses into physical sites by measuring RTTs and adapting the cluster thresholds dynamically. Google's serving infrastructure has grown dramatically in the ten months, and we use our methods to chart its growth and understand its content serving strategy. We find that the number of Google serving sites has increased more than sevenfold, and most of the growth has occurred by placing servers in large and small ISPs across the world, not by expanding Google's backbone.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design; C.4 [**Performance of Systems**]: Measurement techniques

Keywords

CDN, DNS, Geolocation, Measurement

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 *front-end* servers. Clients of these infrastructures are directed to nearby front-ends, which either directly serve static content (e.g., video or images from a content distribution network like Akamai), or use split TCP connections to relay web acccess requests to back-end datacenters (e.g., Google's search infrastructure) [7, 11, 28].

Web service providers employ serving infrastructures to optimize user-perceived latency [31]. They invest heavily in building out these infrastructures and develop sophisticated mapping algorithms to direct clients to nearby front-ends. In recent months, as we discuss later, Google's serving infrastructure has increased sevenfold 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 [2, 14, 26], often focusing on bulk content delivery infrastructures [14], new mapping methodology [2], or new DNS selection methods [26]. In contrast, our work focuses on web serving infrastructures, develops more accurate methods to enumerate and locate front-ends and serving sites, and explores how one infrastructure, Google's, grows over ten months of active buildout.

The first contribution of this paper is a suite of methods to enumerate the IP addresses of front-ends, geolocate them, and cluster them into serving sites. Our methods exploit mechanisms used by serving infrastructures to optimize client-perceived latency. To enumerate the IP addresses, we use the EDNS-client-subnet prefix extension [9] that some serving infrastructures, including Google, use to more accurately direct clients to nearby front-ends. A front-end

[*] 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.

IMC'13 October 23-25, Barcelona, Spain.
Copyright 2013 ACM 978-1-4503-1953-9/13/10
http://dx.doi.org/10.1145/2504730.2504754 ...$15.00.

IP address may sit in front of many physical server machines. In this work, we focus on mapping out the front-end IP addresses, but we do not attempt to determine the number of physical servers. We develop a novel geolocation technique and show that it is substantially more accurate than previously proposed approaches. Our technique, *client-centric geolocation* (CCG), exploits the sophisticated strategies providers use to map customers to their nearest serving sites. CCG geolocates a server from the geographic mean of the (possibly noisy) locations for clients associated with that server, after using speed-of-light constraints to discard misinformation. While EDNS-client-subnet has been examined before [23,26], we are the first to use EDNS-client-subnet to (1) completely enumerate a large content delivery infrastructure; (2) demonstrate its benefit over existing enumeration techniques; and (3) geolocate the infrastructure. We also cluster the front-end IP addresses 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 ten 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 grown sevenfold in the number of front-end sites, with serving sites deployed in over 100 countries and in 768 new ASes. Its recent growth strategy has been to move away from serving clients from front-ends deployed on its own backbone and towards serving from front-ends deployed in lower tiers of the AS hierarchy; the number of /24 prefixes served *off* Google's network more than quadrupled 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 front-ends using client locations alone, but not enough to obviate the need for CCG's filtering techniques.

An explicit non-goal of this work is to estimate the increase in Google's serving capacity: in placing front-ends in ISPs around the world, Google's expansion presumably focused on improving the latency of Web accesses through split-TCP connections [7,11,28], so proximity of front-ends to clients (this paper), and good path performance between clients and front-ends (future work) were more important than capacity increases.

2. BACKGROUND

CDNs and Serving Infrastructures. Adding even a few hundreds of milliseconds to a webpage load time can cost service providers users and business [19,33], 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 [17]. 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 front-end server in the serving site, but the front-end proxies the request back to one of the provider's large datacenters [28]. This arrangement has a number of potential advantages compared to directing the client directly to the datacenter. For example, the client's latency to the front-end is less than the client's latency to the datacenter, allowing TCP to recover faster after loss, the primary cause of suboptimal performance. Moreover, the front-end 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. Of course, the provider will still serve clients near a datacenter directly from that datacenter. But clients in networks that host a serving site are served locally. Front-ends deployed in clients' ISPs usually serve only clients of that ISP (or the ISP's customers), not clients in the ISP's providers or peers.

DNS-based Redirection. Serving infrastructures use the Domain Name System (DNS) to direct clients to appropriate serving sites and front-end servers. When a client queries DNS to resolve a name associated with a service, the service returns an IP address for a front-end 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 [22], enabling services to direct a resolver based on the clients that use it. However, some resolvers serve clients with diverse locations; for these cases no server will be well-positioned for all clients of that resolver.

To overcome this hurdle and provide quality DNS redirections for clients, a number of Internet providers and CDNs proposed EDNS-client-subnet [9]. EDNS-client-subnet is an experimental extension DNS (using its EDNS extension mechanism) allowing clients to include a portion of their IP address in their request. This information passes through possible recursive resolvers and is provided to the authoritative DNS server, allowing a service to select content servers based on the client location, rather resolver location or inferred client location.

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 front-end 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 front-end servers, geolocate serving sites, and cluster front-end servers into serving sites. The challenge in devising these measurement methods is that serving infrastructures are large, distributed entities, with thousands of front-end 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 front-end 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 front-ends. We use this extension to enumerate front-end 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 front-end IP addresses.

To geolocate front-end servers and serving centers, we leverage another mechanism that serving infrastructures have long deployed, namely 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 front-end 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 front-ends into serving sites. We model the relative location of a front-end 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 ten month period, we are able to study Google's serving infrastructure and its evolution. Coincidentally, Google's infrastructure has increased sevenfold 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 front-end 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 front-ends, geolocating them, and clustering them into serving sites.

4.1 Enumerating Front-Ends

Our first goal is to enumerate the IP addresses of all front-ends 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 front-ends is to issue DNS requests from multiple vantage points. Each request returns a front-end 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 mechanism (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 returns a set of front-ends it believes is best suited for clients within the client prefix.

EDNS-client-subnet allows our single measurement site to solicit the recommended front-end for each specified client prefix. Using EDNS-client-subnet, we effectively get a large number of vantage points We query using client prefixes drawn from 10 million routable /24 prefixes obtained Route-Views BGP. Queries against Google using this approach take about a day to enumerate.

4.2 Client-centric Front-End 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 [24] places all Google front-ends in Mountain View, the company's headquarters. (MaxMind may have more accurate locations for IPs belonging to eyeball ISPs, but IPs belonging to transit ISPs will have poor geolocation results.) General approaches such as CBG [12] work best

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

when vantage points are near the target [16], but front-ends in serving infrastructures are sometimes in remote locations, far from public geolocation vantage points. Techniques that use location hints in DNS names of front-ends or routers near front-ends can be incomplete [14].

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 front-ends [35]. 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 front-end, although some clients may be directed to distant front-ends, 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 front-end, (2) query a geolocation database for the locations of those clients, and (3) assume the front-ends 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 front-end, and so may direct it to a distant front-end.
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 front-end that serve them, for a variety of reasons. For example, some front-ends may serve only clients within certain networks, and some clients may have lower latency paths to front-ends other than the nearest ones. In other cases, a serving infrastructure may direct clients to a distant front-end to balance load or may mistakenly believe that the front-end is near the client. Or, a serving infrastructure may not have any front-ends near a particular client.

We now describe how CCG addresses these challenges.

Selecting client prefixes to geolocate a front-end. To enumerate front-ends, CCG queries EDNS using all routable /24 prefixes. However, this approach may not be accurate for geolocating front-ends, for the following reason. Although we do not know the details of how a serving infrastructure chooses which front-end to send a client to, we assume that it attempts to send a client to a nearby front-end 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, Ono [8]. From the user logs we obtain a list of 2.6 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. We emphasize that we use Ono-derived traceroutes to obtain IP prefixes for use with EDNS-client-subnet; other methods for obtaining such prefixes would be equally applicable to our setting, and Ono itself is not necessary for CCG in the sense that we do not make use of the actual platform.

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 inaccurate, based on approaches described in the next paragraph. Second, we combine a large set of client locations to locate each front-end 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 [25] 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. We chose MaxMind because it is freely available and is widely used in research.

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 2.6 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 [13]. Third, it issues ping measurements from all PlanetLab[3] 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 2.6 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 front-ends. Even after filtering bad geolocations, a client may be geographically distant from the front-end it is mapped to, for two reasons: the serving infrastructure may direct clients to distant front-ends for load-balancing, and in some geographical regions, the serving infrastructure deployment may be sparse so that the front-end 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 front-end from all PlanetLab nodes and use the speed of light to establish loose constraints on where the front-end could possibly be [12]. When geolocating the front-end, CCG excludes any clients outside of this region. This excludes 4 million out of 10 million prefixes (1.1 million out of 2.6 million BitTorrent client prefixes). It then estimates the preliminary location for the front-end as the weighted average of the locations of

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

[3] As we show later, we have found that PlanetLab contains a sufficient number of vantage points for speed-of-light filtering to give accurate geolocation.

	10M prefixes	2.6M prefixes
No city-level accuracy	-1.9M (19.5%)	-216K (8.1%)
Multiple locations and client location speed-of-light violations	-8K (.08%)	-2K (.08%)
Front-End location speed-of-light violations	-4M (40%)	-1.1M (41%)
Outside one standard deviation	-392K (3.9%)	-214K (8%)
Remaining	3.7M (37%)	1M (39%)

Table 1: Summary of the number of client prefixes excluded from CCG by filtering. 10M is the 10 million client prefix set and 2.6M is the 2.6 million BitTorrent client prefix set.

	IPs	/24s	ASes	Countries
Open resolver	23939	1207	753	134
EDNS-client-subnet	28793	1445	869	139
Benefit	+20%	+20%	+15%	+4%

Table 2: Comparison of Google front-ends found by EDNS and open resolver. EDNS providers significant benefit over the existing technique.

the remaining client prefixes, then refines this estimate by calculating the mean distance from the front-end to the remaining prefixes, and finds the standard deviation from the mean of the client-to-front-end distances. Our final filter excludes clients that are more than a standard deviation beyond the mean distance to the front-end, excluding 392,668 out of 10 million prefixes (214,097 out of 2.6 million BitTorrent client prefixes).

Putting it all together. In summary, CCG works as follows. It first lists the set of prefixes directed to a front-end, 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 front-end. (Table 1 shows the number of prefixes filtered at each step.) Its preliminary estimate for the front-end 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 front-end as being at the geographic mean of the remaining clients that it serves.

4.3 Clustering front-ends

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 front-end in a higher dimensional metric space, then clustering the front-end in that metric space. Such an approach has been proposed elsewhere [21, 27, 38] and our approach differs from prior work in using better clustering techniques and more carefully filtering outliers.

In our technique, we map each front-end 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 front-ends 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 (a large enough sample size to obtain a robust estimate of propagation latency), and we use the Manhattan distance between two points for clustering (an exploration of other distance norms is left to future work). 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. Despite these heuristic choices, our clustering method works well, as shown in Section 5.3.

The final step is to cluster front-ends 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.

5. VALIDATION

In this section, we validate front-end enumeration, geolocation, and clustering.

5.1 Coverage of Front-End 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 front-ends for a serving infrastructure is to issue DNS queries to the infrastructure from a range of vantage points. Following previous work [14], we do so using open recursive DNS (rDNS) resolvers. We use a list of about 200,000 open resolvers[4]; 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 2 shows the added benefit over rDNS of enumerating Google front-ends using EDNS-client-subnet. Our approach uncovers at least 15-20% more Google front-end IP

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

addresses, prefixes, and ASes than were visible using open resolvers. By using EDNS-client-subnet 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 7 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.

Completeness and EDNS-client-subnet Prefix Length. The choice of prefix length for EDNS-client-subnet queries can affect enumeration completeness. Prefix lengths shorter than /24 in BGP announcements can be 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 subprefixes 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 front-end IP addresses or serving sites. The reply to the EDNS-client-subnet query returns a scope, the prefix length covering the response. Thus, if a query for an IP address in a /24 block returns a scope of, say /25, it means that the corresponding redirection holds for all IP addresses in the /25 covering the query address, but not the other half of the /24. For almost 75% of our /24 queries, the returned scope was also for a /24 subnet, likely because it is the longest globally routable prefix. For most of the rest, we saw a /32 prefix length scope in the response, indicating that Google's serving infrastructure might be doing very fine-grained redirection. We refer the reader to related work for a study of the relationship between the announced BGP prefix length and the returned scope [34].

For our purposes, we use the returned scope as a basis to evaluate the completeness of our enumeration. We took half of the IPv4 address space and issued a series of queries such that their returned scopes covered all addresses in that space. For example, if a query for 1.1.1.0/24 returned a scope of /32, we would next query for 1.1.1.1/32. These brute force measurements *did not uncover any new front-end IP addresses* not seen by our /24 queries, suggesting that the /24 prefix approach in our paper likely provides complete coverage of Google's entire front-end serving infrastructure.

Enumeration Over Time. Front-ends often disappear and reappear from one day to the next across daily enumeration. Some remain active but are not returned in any EDNS-client-subnet requests, others become temporarily inactive, and some may be permanently decommissioned. To account for this variation and obtain an accurate and complete enumeration, we accumulate observations over time, but also test which servers still serve Google search on a daily basis. We check liveness by issuing daily, rate-limited, HTTP HEAD requests to the set of cumulative front-end IP addresses we observe.

The Daily row in Table 3 shows a snapshot of the number of IPs, /24s, and ASes that are observed on 2013-8-8. The Cumulative row shows the additional infrastructure observed earlier in our measurement period but not on that day, and the Inactive row indicates how many of those were not serving Google search on 2013-8-8. This shows that the front-ends that are made available through DNS on a given day is only a subset of what may be active on a given day. For example, for several consecutive days in the first week of August 2013, all our queries returned IP addresses from Google's network, suggesting a service drain of the front-ends in other networks. Our liveness probes confirmed that the majority of front-ends in other networks still actively served Google search when queried, even though no DNS queries directed to them. In the future, we will examine whether we can use our approach to infer Google maintenance periods and redirections away from outages, as well as assess whether these shifts impact performance.

	IPs	/24s	ASes
Daily	22959	1213	771
Cumulative	+5546	+219	+93
–Inactive	-538	-24	-8
Active	27967 (+22%)	1408 (+16%)	856 (+11%)

Table 3: A snapshot from 2013-8-8 showing the differences in number of IPs, /24s, and ASes observed cumulatively across time versus what can be observed within a day. Some front-end IP addresses may not be visible in a daily snapshot. However, IP addresses may be temporarily drained or become permanently inactive or be reassigned. Acquiring an accurate and complete snapshot of active serving infrastructure requires accumulating observations over time and testing which remain active.

5.2 Accuracy of Client-Centric Geolocation

Client-centric geolocation using EDNS-client-subnet shows substantial improvement over traditional ping based techniques [12], undns [32], and geolocation databases [24].

Dataset. To validate our approach, we use the subset of Google front-ends with hostnames that contain airport codes hinting at their locations. Although the airport code does not represent a precise location, we believe that it is reasonable to assume that the actual front-end is within a few 10s of kilometers of the corresponding airport. Airport codes are commonly used by network operators as a way to debug network and routing issues so having accurate airport codes is an important diagnostic tool. Previous work has show that only 0.5% of hostnames in a large ISP had misleading names [39], and so we expect that misnamed Google front-ends only minimally distort our results. A limitation of our validation is that we cannot validate against Google hosted IPs that do not have airport codes because popular geolocation databases such as MaxMind place these IPs in Mountain View, CA. Using all 550 front-ends with airport codes, we measure the error of our technique as the distance between our estimated location and the airport location from data collected on April 17, 2013.

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 [12], a technique that issues traceroutes to front-ends and locates the front-ends based on geographic hints in names of nearby routers [14], and the MaxMind GeoLite Free database [24]. 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 front-ends, respectively. CBG performs well when vantage points are close to the front-end [16], but it incurs large errors for the half of the front-ends in more remote regions. The traceroute-based technique is unable to provide any location for 20% of the front-ends because there were no hops with geographic hints in their hostnames near to the front-end. The MaxMind database performs poorly because it places most front-ends belonging to Google in Mountain View, CA.

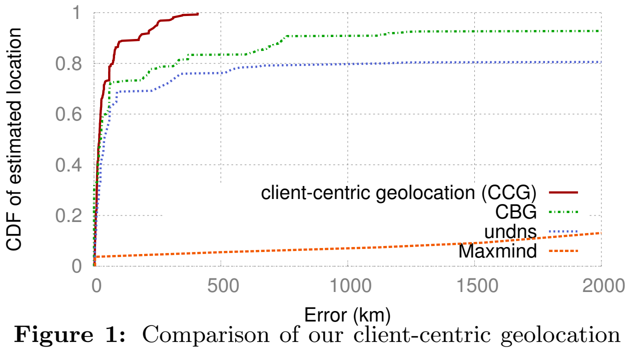

Figure 1: Comparison of our client-centric geolocation against traditional techniques, using Google front-ends with known locations as ground truth.

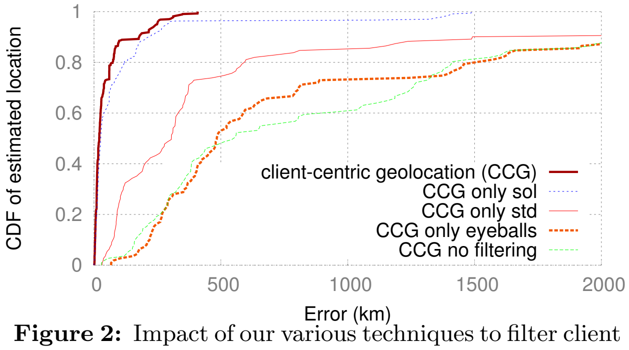

Figure 2: Impact of our various techniques to filter client locations when performing client-centric geolocation on Google front-ends with known locations.

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 front-ends 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 front-end and to exclude clients with infeasible MaxMind locations, we obtain a median error of 26km, and only 10% of front-end 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 front-ends in Stockholm, Sweden, with the *arn* airport code, serving a large group of client locations throughout Northern Europe. However, our technique locates the front-ends 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 front-end. 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 front-ends in the 24ms range, yielding a feasible radius of 2400km. This loose constraint results in poor geolocation for this set of front-ends.

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 datacenters 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.

We also found that there are instances where we are unable to place a front-end. In particular, we observed that occasionally when new front-ends were first observed during the expansion, there would be very few /24 client networks directed to them. These networks may not have city-level geolocation information available in MaxMind so we were unable to locate the corresponding front-ends.

5.3 Accuracy of Front-End 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; and finally, 75 Google front-end IP addresses, that have airport codes in their reverse DNS names, out of 550 (14%) having airport codes and of 8,430 (0.9%) total Google IP addresses as of Apr 16th, 2013. 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. In the absence of complete ground truth, we have had to rely on more approximate validation techniques: using PlanetLab, selecting a subset of front-ends with known locations, and using airport codes (we have justified this choice in Section 5.2). We also discuss below some internal consistency checks on our clustering algorithm that gives us greater confidence in our results.

The metric we use for the accuracy of clustering is the *Rand Index* [29]. 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.

Experiment	Rand Index	False negative	False positive
PlanetLab	0.99	1%	0
CA	0.97	0	3%
Google	0.99	1%	0

Table 4: Rand index for our nodesets. Our clustering algorithm achieves over 97% across all nodesets, indicating very few false positives or negatives.

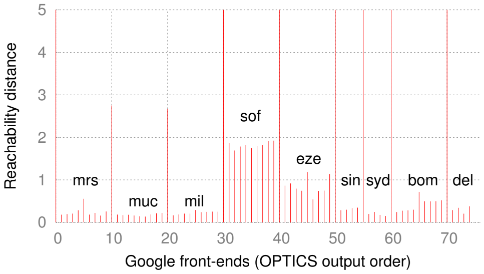

Figure 3: Distance plot of Google servers with airport codes. Servers in the same cluster have low reachability distance to each other thus are output in sequence as neighbors. Cluster boundaries are demarcated by large impulses in the reachability plot.

Table 4 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 positives and false negatives. In the California nodeset, the method fails to set apart USC/ISI nodes from nodes on the USC campus (10 miles away, but with the same upstream connectivity) which leads to 3% false positive. In the Planet lab nodeset, some clusters have low reachability distance that confuses our boundary detection method, resulting in some clusters being split into two. 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 front-ends in one serving site have a prefix `mrs02s04`, and all front-ends in the other serving site have a prefix `mrs02s05`.

In addition, Figure 4 shows the OPTICS output when using reverse-TTL (as proposed in [21]) instead of RTT for the metric embedding. This uses the same set of Google servers as in our evaluation using RTT for metric embedding. We could see that 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. 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).

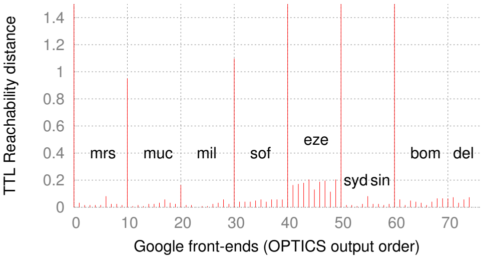

Figure 4: The output of the OPTICS clustering algorithm when reverse-TTL is used for the metric embedding. When using this metric, the clustering algorithm cannot distinguish serving sites at Bombay (`bom`) and Delhi (`del`) in India, while RTT-based clustering can.

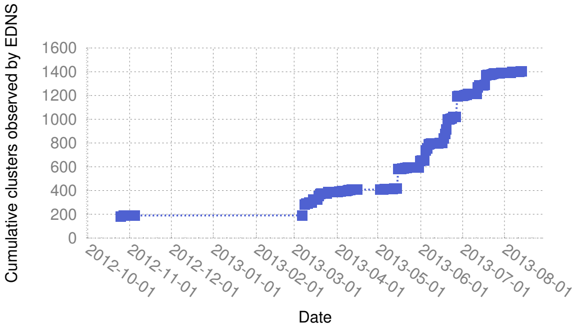

Figure 6: Growth in the number of points of presence hosting Google serving infrastructure over time.

Finally, we also perform some additional consistency checks. We run our clustering algorithm against all Google front-end IPs that have airport codes (6.5%, 550 out of 8430). We find that except for the kind of false negative we mentioned above (multiple serving site within same airport code), the false positive rate of our clustering is 0, which means we never merge two different airport codes together. Furthermore, when our algorithm splits one airport code into separate clusters, the resulting clusters exhibit naming consistency — our algorithm always keeps IPs that have same hostname pattern <airport code><two digit>s<two digit>, such as `mrs02s05`, in the same cluster.

In summary, our clustering method exhibits over 97% accuracy on three different test datasets. On the Google IPs that have airport codes, our clustering show one kind of false negative that we believe to be correct and no false positive at all.

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 through August of 2013. We are able to capture a substantial expansion of Google infrastructure.

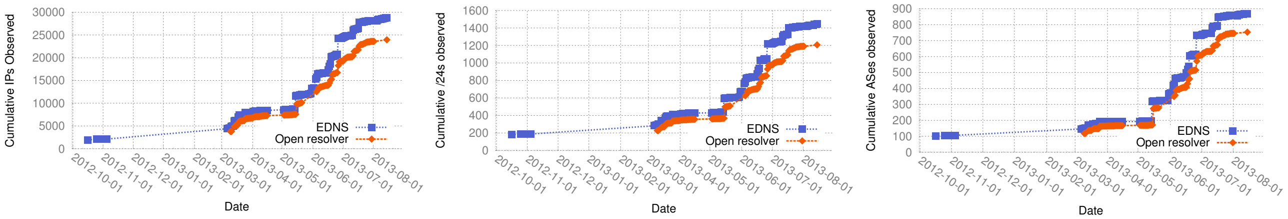

Figure 5: Growth in the number of IP addresses (a), /24 prefixes (b), and ASes/countries (c) observed to be serving Google's homepage over time. During our study, Google expanded rapidly at each of these granularities.

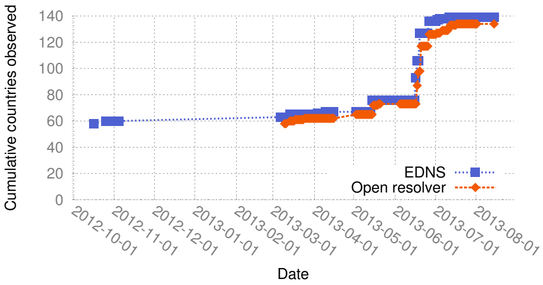

Figure 8: Number of countries hosting Google serving infrastructure over time.

6.1 Growth over time

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.[5] The graph shows slow growth in the cumulative number of Google IP addresses observed between November 2012 and March 2013, then a major increase in mid-March in which we saw approximately 3,000 new serving IP addresses come online. This was followed by another large jump of 3,000 in mid-May. Over the month of June, we observed 11,000 new IPs followed by an increase of 4,000 across July.

By the end of our study, the number of serving IP addresses increased sevenfold. 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 8X 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 front-end IP addresses that appeared during the course of our study, 95% are in ASes other than Google. Of those addresses, 13% are in the United States and 26% are in Europe, places that would appear to be well-served directly from Google's network.[6] In the future, we plan to investigate the performance impact of these front-ends. In addition, 21% are in Asia, 13% are in North America (outside the US), 11% are in South America, 8% are in Africa, and 8% are in Oceania. A link to an animation of the worldwide expansion is available at http://mappinggoogle.cs.usc.edu.

Figure 8 depicts this growth in the number of countries hosting serving infrastructure, from 58 or 60 at the beginning of our study to 139 in recent measurements.[7] 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 5 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 [37],[8] except that we only classify as stubs ASes with 0 customers, and we introduce a Tiny ISP class for ASes with 1-4 customers.

As seen in the table, the rapid growth in ASes that 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 8% of them to servers off its network, representing a 393% increase in the number served from outside 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 [7,11,28], 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 front-ends we located outside of Google's network. More than 20% 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.[9] It is

[5] It is not necessarily the case that each IP address maps to a distinct front-end.

[6] In contrast, when we submitted the paper in May, only 13% were in the US or Europe. We added in the new expansion in those regions in preparing the final version of the paper.

[7] We base our locations on our CCG approach, which may distort locations of front-ends that are far from their clients.

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

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

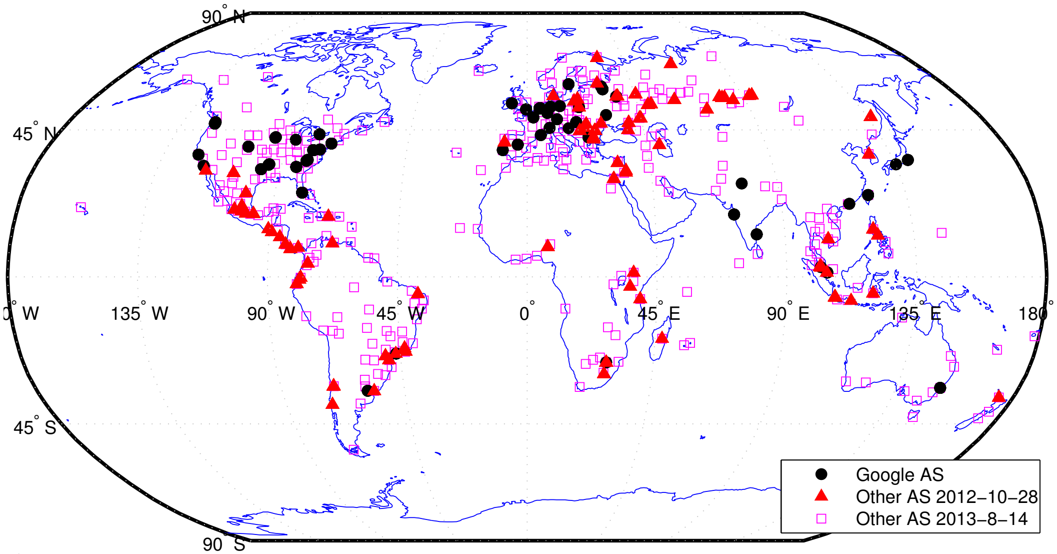

Figure 7: A world wide view of the expansion in Google's infrastructure. Note that some of the locations that appear floating in the Ocean are on small islands. These include Guam, Maldives, Seychelles, Cape Verde and Funchal.

	November 2012		May 2013				August 2013			
	ASes	Clients	ASes		Clients		ASes		Clients	
Google	2	9856K	2	(+0%)	9658K	(-2%)	2	(+0%)	9067K	(-8%)
Tier 1	2	481	2	(+0%)	201	(-58%)	4	(+100%)	35K	(+7278%)
Large	30	111K	46	(+53%)	237K	(+114%)	123	(+310%)	410K	(+270%)
Small	35	37K	64	(+83%)	63K	(+71%)	319	(+811%)	359K	(+870%)
Tiny	23	31K	41	(+78%)	57K	(+84%)	206	(+796%)	101K	(+228%)
Stub	13	21K	36	(+177%)	38K	(+81%)	201	(+1446%)	79K	(+281%)

Table 5: Classification of ASes hosting Google serving infrastructure at the beginning, middle, and end of our study. We count both the number of distinct ASes and the number of client /24 prefixes served. Growth numbers for May and August are in comparison to November. Google still directs 90% of the prefixes to servers within its own network, but it is evolving towards serving fewer clients from its own network and more clients from smaller ASes around the world.

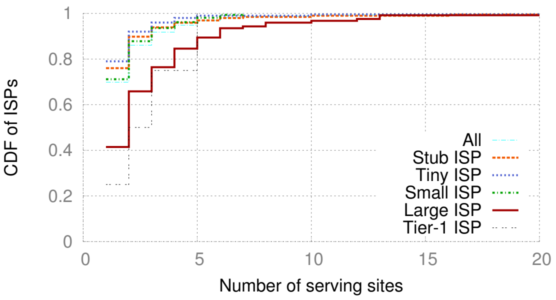

Figure 9: CDF of number of sites in different types of ISP.

possible that the servers were already in use as video caches; if so, this existing physical deployment could have enabled the rapid growth in front-ends we observed.

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. Overall, nearly 70% of the ISPs host only one serving site. Generally speaking, smaller ISPs host fewer serving sites than larger ISPs. The biggest exceptions are a Tiny ISP in Mexico hosting 23 serving sites consisting of hundreds of front-end IPs, and a Stub national mobile carrier with 21 sites. Befitting their role in the Internet, most Large and Tier 1 ISPs host multiple sites. For example, a Large ISP in Brazil serves from 23 sites.

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, 93% are located within the server's AS's customer cone (the AS itself, its customers, their customers, and so on) [20]. Since correctly inferring AS business relationship is known to be a hard problem [10], it is unclear whether the remaining 7% 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 40% of the non-customer cases stem from just 7 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 beginning of the study. Figure 10(a) shows the distribution of the distance from a client to our estimate of the location

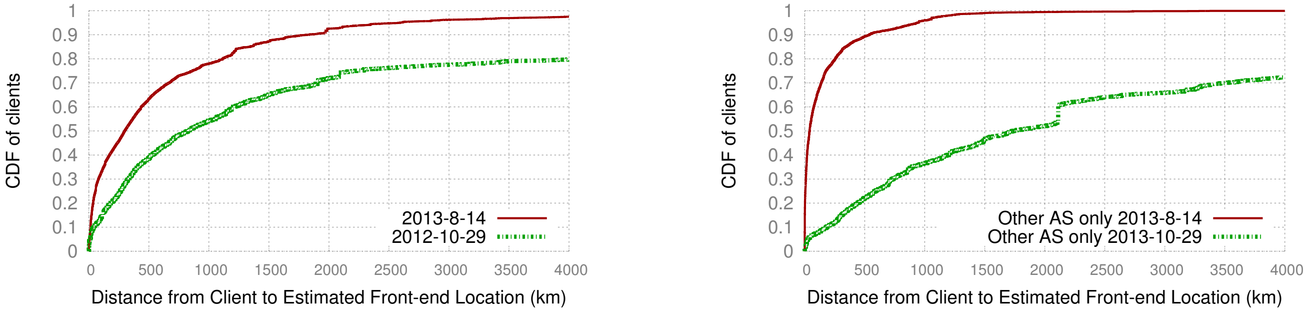

Figure 10: (a) Distances from all BitTorrent client prefixes to estimated front-end locations to which Google directs them. **(b)** Comparison of the distances between the set of clients served by front-ends outside of Google's network on 2013-8-14 and their estimated front-end locations on 2013-8-14 and 2012-10-29.

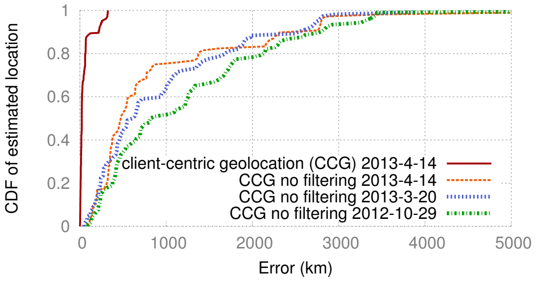

Figure 11: As Google expands, clients become closer to their front-ends, improving accuracy of filter-less client-based geolocation.

of the server serving it. We restrict the clients to those in our BitTorrent eyeball dataset (2.6 million client prefixes) 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-August 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 front-ends increases from 39% to 64%, and the fraction within 1000km increases from 54% to 78%. Figure 10(b) shows the distribution of distances only for the set of client prefixes that were directed to front-ends outside of Google's network on 2013-8-14. The top curve shows the distances between the clients and front-ends on 2013-8-14 while the bottom curve shows the distances between this same set of clients and the front-ends that they were served by on 2012-10-29. The figure shows that the set of clients that have moved off of Google's network are now much closer to their front-ends in August of 2013 than in October of 2012. The fraction of client prefixes within 500km of their front-ends has increased from 21% to 89%, and the fraction within 1000km increased from 36% to 96%. Because many of the newer front-ends 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 front-end will have on application performance [28].

6.3 Impact on Geolocation Accuracy

A side-effect of Google directing more clients to front-ends closer to them is that our geolocation technique should become more accurate over time, since we base it on the assumption that front-ends 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. 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. Our data is publicly available at http://mappinggoogle.cs.usc.edu.

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 front-ends; (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 front-end if the client is a customer of the front-end'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 front-ends).

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 [35]; we believe our mapping can improve geolocation, peer selection, and AS classification.

Geolocation is a much studied problem [12, 13, 16], 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 [8]. However, with the existing plugin, the client can only assess similarity of other users of the plugin who send their CDN front-end 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 [15] or with the customer cone dataset we used [20]. 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-front-end data we generate could help resolve ambiguities in AS relationships and lead to better inference in the future.

Mapping Other Providers. While our techniques will apply directly for some providers, we will need to adapt them for others, and we describe the challenges and potential approaches here. Our studies of Google combine observations using EDNS-client-subnet and open recursive resolvers. EDNS-client-subnet support is increasing. However, some networks such as Akamai do not support it, and we are restricted to using open resolvers for them.

In Section 5.1, we demonstrated that even using hundreds of thousands of open DNS resolvers would miss discovering much of Google's infrastructure. Table 2 showed that EDNS-client-subnet found 20% more front-end IPs than open resolvers, but we cannot assume that ratio holds on other infrastructures. We would expect open resolvers to suffice to uncover all of a ten-front-end infrastructure, for example, but we would expect an even bigger gap on Akamai than on Google, since Akamai serves from many more locations.

We may be able to use our results from Google to project results for other providers that support only open resolvers.

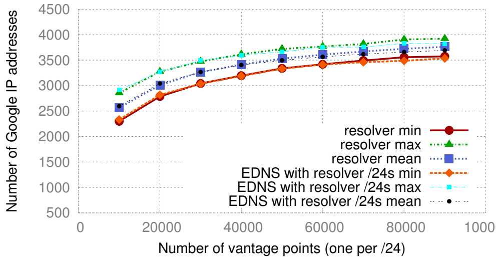

Figure 12: The relation between number of Google IP addresses discovered and the number of vantage points. Using one open resolver per /24 block and one EDNS query per /24 block.

We select one open recursive resolver from each /24 in which we know one (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 query for that prefix. Figure 12 depicts the growth in the number of Google front-end IP addresses discovered by the two approaches as we issue additional measurements (1000 trials). Using resolvers in a set of prefixes yields very similar results to issuing EDNS queries for that same set of prefixes, so that the benefit of EDNS is primarily that we can issue queries for many more prefixes than we have access to resolvers in.

We 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 front-end servers (as of May 4th, 2013). Using EDNS-client-subnet queries for these 10M prefixes, we found 8563 IP addresses, within the range, so 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. We can use our Google results to characterize which regions our set of open resolvers has good coverage in, in order to flag portions of other infrastructures as more or less complete.

Google lends itself to client-centric geolocation because our EDNS-client-subnet measurements uncover the client to front-end mappings, and Google's deployment is dense enough that most clients are near front-ends. We will have to adapt the strategy for deployments where one or both of those properties do not hold. Akamai uses a resolver's geolocation in mapping decisions [5], and so it may be possible to geolocate Akamai servers based on the locations of the open resolvers they serve, even though we cannot directly measure which clients they serve. We will verify the soundness of this approach by geolocating Google front-ends using resolver locations. If the approach is generally accurate, we can also use it to flag suspicious resolver locations and only use the remainder when geolocating Akamai (or other) servers.

CDNs such as Edgecast support EDNS queries to discover client-to-front-end mappings, but they lack the density of servers of Akamai and Google and so necessarily direct some clients to distant servers. Since our geolocation approach assumes front-ends are near clients, it may not be sound to assume that the front-end is at the geographic center of the clients. Edgecast publishes its geographic points of presence on its website, so we can use its deployment as ground truth to evaluate approaches to map other providers that do not publish this information. We will investigate whether our aggressive pruning of distant clients suffices for Edgecast. If not, straightforward alternate approaches may work well for these sparse deployments. For example, these small deployments tend to be at well-connected Internet exchange points, where we likely have a vantage point close enough to accurately use delay-based geolocation [12, 16].

8. RELATED WORK

Closest to our work is prior research on mapping CDN infrastructures [1,2,14,36]. Huang et al. [14] map two popular content delivery networks, Akamai and Limelight, by enumerating their front-ends using a quarter of a million open rDNS resolvers. They geolocate and cluster front-ends using a geolocation database and also use the location of penultimate hop of traceroutes to front-ends. Ager et al. [2] chart web hosting structures as a whole. They start by probing several sets of domain names from dozens of vantage points to collect service IP addresses, rely entirely on MaxMind [24] for geolocation, and use feature-based clustering where the goal of clustering is to separate front-ends belonging to different hosting infrastructures. Torres et al. [36] 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 to achieve complete coverage, 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.

In addition to this prior work, simultaneous work appearing at the same conference as this paper also used EDNS-client-subnet to expose CDN infrastructure [34]. While our work focuses on characterizing Google's rapid expansion, including geolocating and clustering front-ends, that work addresses complementary issues including measuring other EDNS-client-subnet-enabled infrastructures. Our results differ from that work, as our work exposes 30% more /24 prefixes and 12% more ASes hosting front-ends that are actively serving Google search. We believe our additional coverage results from our more frequent mapping and accumulation of servers over time, since a single snapshot may miss some infrastructure (see Table 3). Some in the operations community also independently recognized how to use EDNS-client-subnet to enumerate a CDN's servers, although these previous measurements presented just a small-scale enumeration without further investigation [23].

Several other pieces of work are tangentially related to ours. Previous work exploits the observation that two clients directed to the same or nearby front-ends are likely to be geographically close [8, 35]. Our work uses this observation to geolocate front-ends. Mao et al. [22] quantify the proximity of clients to their local DNS resolvers and find 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. Otto et al. [26] examine the end-to-end impact that different DNS services have on CDN performance. It is the first work to study the potential of EDNS-client-subnet to address the client CDN mapping problem, using the extension as intended, but does not repurpose EDNS to map infrastructure, as we do.

Finally, several strands of research explored complementary problems associated with serving infrastructures, including characterizing and diagnosing latency of providers [7,11,17,18,40]; geolocating ASes using client locations [30]; verifying data replication strategies for cloud providers [4]; and analyzing content usage in large CDNs [6]. Some of this research describes how providers use distributed front-ends as proxies to improve client performance [7, 11, 28]. Our work demonstrates the rapid expansion–and new strategy of front-ends in other networks–of Google's infrastructure to delivery on this approach.

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 world 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 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 grew more than sevenfold, and we see Google deploying satellite front-ends 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.

Acknowledgments

We thank our shepherd, Aditya Akella, and the anonymous IMC reviewers for their valuable feedback. We also gratefully acknowledge Bernhard Ager, Georgios Smaragdakis, Florian Streibelt, and Nikolaos Chatzis for their feedback on earlier versions of this paper.

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. Matt Calder and Ramesh Govindan were partially supported by the U.S. National Science Foundation grant number CNS-905596.

10. REFERENCES

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

[2] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. Web content cartography. In *IMC*, 2011.

[3] Mihael Ankerst, Markus M. Breunig, Hans-peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. In *SIGMOD*, 1999.

[4] Karyn Benson, Rafael Dowsley, and Hovav Shacham. Do you know where your cloud files are? In *Cloud Computing Security Workshop*, 2011.

[5] Arthur Berger, Nicholas Weaver, Robert Beverly, and Larry Campbell. Internet nameserver IPv4 and IPv6 address relationships. In *IMC*, 2013.

[6] 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 *IMC*, 2007.

[7] 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 *IMC*, 2011.

[8] David Choffnes and Fabian E. Bustamante. Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *SIGCOMM*, 2008.

[9] 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).

[10] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, k. c. claffy, and George Riley. AS relationships: Inference and validation. *ACM CCR*, 37(1):29–40, January 2007.

[11] Tobias Flach, Nandita Dukkipati, Andreas Terzis, Barath Raghavan, Neal Cardwell, Yuchung Cheng, Ankur Jain, Shuai Hao, Ethan Katz-Bassett, and Ramesh Govindan. Reducing web latency: the virtue of gentle aggression. In *SIGCOMM*, 2013.

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

[13] Zi Hu and John Heidemann. Towards geolocation of millions of IP addresses. In *IMC*, 2012.

[14] 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.

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

[16] 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*, 2006.

[17] 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*, pages 190–201, 2009.

[18] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. CloudCmp: comparing public cloud providers. In *IMC*, 2010.

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

[20] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, and k claffy. AS relationships, customer cones, and validation. In *IMC*, 2013.

[21] 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.

[22] 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. In *USENIX Annual Technical Conference*, 2002.

[23] Mapping CDN domains. `http://b4ldr.wordpress.com/2012/02/13/mapping-cdn-domains/`.

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

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

[26] John S. Otto, Mario A. Sánchez, John P. Rula, and Fabián E Bustamante. Content delivery and the natural evolution of DNS. In *IMC*, 2012.

[27] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *SIGCOMM*, 2001.

[28] 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.

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

[30] Amir H Rasti, Nazanin Magharei, Reza Rejaie, and Walter Willinger. Eyeball ASes: from geography to connectivity. In *IMC*, 2010.

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

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

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

[34] Florian Streibelt, Jan Böttger, Nikolaos Chatzis, Georgios Smaragdakis, and Anja Feldmann. Exploring EDNS-client-subnet adopters in your free time. In *IMC*, 2013.

[35] Ao-Jan Su, David R. Choffnes Aleksandar Kuzmanovic, and Fabi'an E. Bustamante. Drafting behind Akamai (Travelocity-based detouring). In *SIGCOMM*, 2006.

[36] 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 *ICDCS*, 2011.

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

[38] Qiang Xu and Jaspal Subhlok. Automatic clustering of grid nodes. In *Proc. of 6th IEEE International Workshop on Grid Computing*, 2005.

[39] Ming Zhang, Yaoping Ruan, Vivek S Pai, and Jennifer Rexford. How DNS misnaming distorts Internet topology mapping. In *USENIX Annual Technical Conference*, 2006.

[40] 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.