Technical Report
Number: NU-EECS-16-11

June, 2016

Fury Route: Leveraging CDNs to Remotely Measure Network Distance

Marcel Flores, Alexander Wenzel, Kevin Chen, Aleksandar Kuzmanovic

Abstract

Estimating network distance between arbitrary Internet endpoints is an essential primitive in networking. We show that the heterogeneous infrastructure of different CDNs, combined with the proliferation of the EDNS0 client-subnet extension (ECS), enables novel infrastructureless measurement. We design Fury Route, a system that estimates network distance by utilizing ECS to construct a virtual path between endpoints via intermediate CDN replicas.

Keywords

Network distance, Content distribution networks, Internet measurement

Fury Route: Leveraging CDNs to Remotely Measure Network Distance

Marcel Flores
Northwestern University

Alex Wenzel
Northwestern University

Kevin Chen
Northwestern University

Aleksandar Kuzmanovic
Northwestern University

Abstract

Estimating network distance between arbitrary Internet endpoints is an essential primitive in applications ranging from performance optimization to network debugging and auditing. Enabling such a primitive without deploying new infrastructure was demonstrated via DNS. However, the proliferation of DNS hosting has made DNS-based measurement techniques far less dependable. In this paper, we show that the heterogeneous infrastructure of different CDNs, combined with the proliferation of the EDNS0 client-subnet extension (ECS), enables novel infrastructureless measurement. We design Fury Route, a system that estimates network distance by utilizing ECS to construct a virtual path between endpoints via intermediate CDN replicas.

Fury Route requires no additional infrastructure to be deployed. The measured endpoints do not need to participate by sending or responding to probes. Fury Route further generates no load on endpoints. It only queries DNS, whose infrastructure is designed for large loads. We extensively evaluate Fury Route and demonstrate that (*i*) the key to Fury Route's ability to construct virtual paths lies in the heterogeneity of the underlying CDNs, (*ii*) Fury Route is effective in revealing relative network distance, needed in many real-world scenarios, (*iii*) caching can dramatically reduce Fury Route's DNS overhead, making it a useful system in practice.

1. INTRODUCTION

The ability to estimate network distance between arbitrary endpoints on the Internet is fundamentally necessary in numerous scenarios [24]. Such estimates have been shown to heavily correlate with actual end-to-end performance (in terms of throughput and delay) between the two endpoints [32, 39].

In [24], Gummadi *et al.* showed that DNS infrastructure could be effectively utilized to measure network distance without access to any of the endpoints. By using open recursive DNS resolvers and by relying on the proximity of clients and servers to their authoritative DNS servers, they manage to approximate the distance between the endpoints. Nonetheless, 15 years later, the Internet has become a much different place. On one hand, the number of open recursive DNS resolvers is rapidly decreasing [26]. On the other hand, DNS hosting (*i.e.*, outsourcing DNS services to the cloud [1, 2, 3, 4, 5, 6, 7, 9, 10, 12, 14, 15]), is fundamentally blurring the assumption of co-location of endpoints (both clients and servers) and authoritative DNS servers. In our experiments, we find that only 4% of internet hosts had DNS servers which responded to outside recursive queries, severely limiting the applicability of King. This trend is only going to evolve in future, making DNS-based latency measurement techniques far less dependable.

In this paper, we present Fury Route, a system that aims to estimate the network distance between arbitrary Internet endpoints. It does so by relying on network features which are likely to grow more pronounced. In particular, Fury Route relies on (*i*) the existence of different Content Distribution Networks (CDNs) and their heterogeneity in terms of the number and deployment of their replicas, (*ii*) CDNs' common desire to direct clients to nearby CDN replicas, and (*iii*) the proliferation of EDNS0 client-subnet extension (ECS) [20], a mechanism by which a host issuing DNS requests can indicate the origin of the request. In a nutshell, *Fury Route constructs a virtual path between source and destination, consisting of CDN replicas from different providers, by issuing ECS requests on behalf of endpoints and intermediate CDN replicas on the path.* We show that the length of such a constructed path correlates with the latency between the two endpoints.

Fury Route requires no additional infrastructure to be deployed. The measured endpoints do not need to cooperate, neither by sending nor responding to any probes. Fury Route further generates no load on the parties involved. It only queries DNS, whose infrastructure is designed to handle large loads. Moreover, while Fury Route utilizes the DNS infrastructure, it is in no way impacted by the number or availability of recursive DNS resolvers, nor is it affected by DNS hosting. Fury Route utilizes the significant mapping work done by various CDNs, and it effectively extracts this information via DNS.

To begin, Fury Route initially issues ECS requests on behalf of the source and destination for domains served on multiple CDNs. Then, it constructs a path towards the destination by generating an iterative series of ECS queries, which "hop" between intermediate CDN replicas by issuing new requests on behalf of these CDN replicas. The key to Fury Route's path construction is its ability to move "closer" to the destination in each iteration, as determined by the overlap of CDN replicas seen by the current hop and the destination. As different CDNs have different deployments and granularity (*e.g.*, large-scale fine-grained Google CDN *vs.* coarse-grained Alibaba CDN), they offer differing views of the underlying network. We demonstrate that this heterogeneity allows Fury Route to connect arbitrary endpoints on the Internet.

In our evaluation, we show that ECS is widely deployed on the Internet today, with 86% of pages in the Alexa Top 500 contacting at least one domain which supports ECS. We further evaluate the Fury Route system, and find that in the median case, it is able to construct chains between 90% of origin and destination pairs. We further demonstrate that it is able to correctly order up to 83% of destinations in the median case. We explore the effectiveness of caching our paths, demonstrating that a well formed cache can reduce the overhead of 87% of paths to the minimum possible. Finally, we examine the effects of shifting CDN behaviors and demonstrate that a single snapshot provides reliable performance.

Necessarily, Fury Route is not perfect, and comes with limitations and caveats. First, due to the nature of virtual end-to-end path design, Fury Route isn't designed to measure absolute latency, a process that is prone to inevitable imprecision. Nonetheless, we demonstrate that Fury Route is highly effective in determining *relative network distance* information. Many real-world applications, including peer or server selection as well as Web performance and auditing, rely only on relative network distance. Second, Fury Route fundamentally depends on the underlying CDN infrastructure to provide estimates. Consequently, its performance, both in its ability to construct the chains and to estimate network distance, is necessarily better in regions where CDN deployment is richer. Third, the volume of DNS requests Fury Route must send necessarily depends on the number of underlying CDNs used. We demonstrate in Section 5.5 that caching can dramatically reduce the number of DNS requests generated by Fury Route.

We make the following contributions:

- We demonstrate that the proliferation of ECS, a feature designed to enable a more efficient user mapping by CDNs, effectively turns the DNS into an open database of CDNs' user mappings that could be used for remote network distance estimation.
- We design Fury Route, a system that queries this database to construct a virtual path between arbitrary Internet endpoints, and estimate network distance between them.
- We evaluate Fury Route and demonstrate that it is a practical system capable of estimating network distance without introducing any new infrastructure nor without generating any probing traffic.

This paper is structured as follows. Section 2 provides large-scale measurements to characterize the real-world deployment and properties of ECS in today's Internet. In Section 3, we present Fury Route's design, and its implementation in Section 4. Section 5 provides Fury Route's evaluation and we provide a discussion in Section 6. We present related work in Section 7 and conclude in Section 8.

2. BACKGROUND AND MEASUREMENT

Before we present the design of Fury Route, we first motivate the need for remote measurements. We then consider the nature of the EDNS0 client-subnet extension and the features which can be of the most meaningful use to Fury Route. We then consider a set of measurements conducted to understand the nature of ECS in the "wild," as well as the nature of ECS responses.

2.1 The Need for Remote Measurements

Despite the prevalence of large scale networks with significant visibility of the network, the need for measurements between *remote* hosts on the Internet remains. Beyond classical applications, such as peer or server selection, the estimates of network distance between arbitrary Internet endpoints is needed in the construction of topology-aware multicast overlays [24]. Moreover, it significantly simplifies the execution of wide-area measurement studies at scale [24]. Remote network mesurements are further very useful in emerging networking scenarios as well. For example, modern Web pages are characterized by a high degree of content fragmentation, *i.e.*, pieces of content on a Web page (ads, images, *etc*) are served by numerous different providers, servers, CDNs, and locations [29]. In such a scenario, the site operator has control of neither the client nor the provider. Here the site operator must at least detect potentially very distant low-performance providers, yet with no direct ability to perform measurements between the providers and clients. The ability to estimate performance between clients and providers is needed not just for performance optimization but also for auditing and debugging purposes.

Figure 1 presents the fundamental pieces of such a scenario. Since the client and provider are both out-

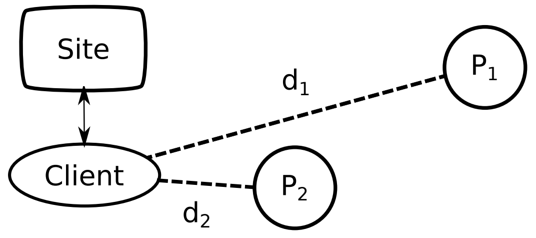

Figure 1: An example of a scenario in which a web site operator must choose between providers (P_1 and P_2) for a client with no direct measurement ability.

side the control of the site operator no direct measurement can be done. However, the operator must avoid a provider with extensively-long network distance from the client. Fury Route provides recourse in such a scenario, allowing the operator to estimate the relative difference between the client and the providers, *e.g.*, detect that d_1 is longer than d_2. As it stands, Fury Route is the only system that provides an infrastructure-free mechanism to estimate distance between remote hosts in such scenarios.

2.2 EDNS0 Client-Subnet

The EDNS0 `client-subnet` extension (ECS) provides a mechanism by which a host issuing DNS requests can label their requests with a subnet, indicating the origin of the request. The purpose of this extension is to aid in DNS-based replica selection and addresses challenges which arise from clients being far away from their LDNS server [20, 31]. When providing a subnet, the client also indicates a subnet length, allowing precise control of how specific the request is. This enables clients to explicitly adjust the balance between accuracy and privacy.

Upon receiving an ECS request, the authoritative DNS server uses the submitted subnet to perform its replica selection, according to its individual policy. When responding to the query, the answer includes a scope netmask field. If this value is less than or equal to the client-specified subnet length(*i.e.*, a larger subnet), it indicates the set of subnets which would receive the same result, for caching purposes. If the value is greater than the supplied length (*i.e.*, smaller subnet), it indicates the DNS server would like the client to resubmit with a more specific subnet.

Fury Route will take advantage of EDNS0 in two ways. First, it uses the `client-subnet` field to send requests from arbitrary locations, granting it a wide view of provider replicas from anywhere in the entire Internet. Second, it exploits the value of the scope netmask in the response in order to understand the *quality* of the set of responses. While these actual values are likely a function of each network's particular layout, policy, and current load, they still provide feedback on how well the provider was able to match a particular client subnet.

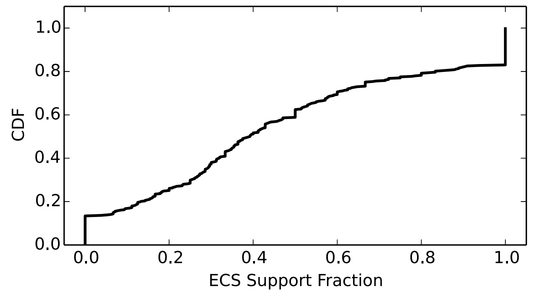

Figure 2: A CDF of the fraction of contacted domains per page which support EDNS over the Alexa Top 500.

2.3 Observed Networks

As Fury Route is built on the efforts and deployments of existing networks, we present a brief measurement study of the networks which we use and the behaviors that they exhibit. While similar in mechanism, the measurement study presented here varies fundamentally from previous work [17, 37]. In particular, while the size of a network is important, we are chiefly interested in the character of ECS responses, rather than their actual content. Specifically, we are concerned with the granularity, consistency, and location variety of responses, as such features will be critical for Fury Route.

First, we examine the prevalence of sites which respond to ECS queries. We load each landing page on the Alexa Top 500 and record all DNS traffic which occurs during each load. This includes all DNS requests sent as the result of third-party providers such as Ad networks and CDNs and resulted in 10, 013 total DNS names. We further select IP addresses from 25 Planet Lab nodes, distributed around the world, with 5 nodes from each continent, excluding Africa. Next, for each domain name in our set, we issue 25 ECS queries to Google DNS (from a single machine), using each of the 25 Planet Lab IP addresses as the client-subnet value with a full /32 prefix. This allows us to see *(i)* the set of providers which support ECS, *(ii)* which providers are performing client mapping in their responses, *(iii)* a coarse estimate of the size of each provider network.

Figure 2 presents a CDF of the number of domains loaded by each page which support ECS, based on the above criteria. We see that 14% of pages encountered no ECS domains. On the other extreme, 18% of pages consist entirely of ECS supporting providers. In the median case, 40% of domains offer ECS support. Example Alexa 500 pages that fully support ECS are `google.com`, `gmail.com`, `youtube.com`, `mozilla.org`, `tumblr.com`, `feedly.com`. Example Alexa 500 pages that provide no open support for ECS, *i.e.*, the 14% shown in Figure 2, are mostly Akamai-supported pages, such as `adobe.com`, `apple.com`, and `bbc.co.uk`. While Akamai does support ECS [18], it does so only via OpenDNS [13]

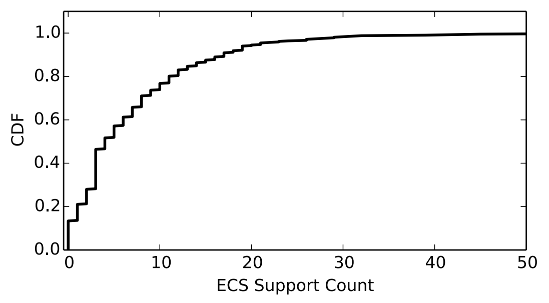

Figure 3: A CDF of the number of contacted domains per page which support EDNS over the Alexa Top 500.

and GoogleDNS [11] public DNS services, using the actual IP address of the *requester*. As such, Akamai's CDN is currently not directly usable by Fury Route, as Fury Route requires sending ECS requests on behalf of remote IPs. Consequently, the ECS deployment shown in Figure 2 presents a strict lower bound.

Figure 3 offers further insight, showing a CDF of the number of ECS supported domains per page. Here, we see that the median case contacts 5 domains with ECS support, but many pages contact significantly more. These results may depend on the nature of each site operator's relationship with their third party providers: they may contract with several CDNs and may have a complex blend of configurations with those providers. In total, we observed $1,749$ total unique hostnames which indicated support for ECS. While this analysis suggests broad deployment of ECS, Fury Route will depend on having a variety of providers, as well as distributed and diverse infrastructure from each provider. We therefore explore the nature of ECS responses when considered as *providers*, rather than simply individual hostnames.

In examining our set of $1,749$ ECS supporting hostnames, we noted that many resolved to matching canonical names, while others did not. Therefore, in order to discern between providers, we chose to pursue an approach based on the A records in observed responses. We resolved each of these hostnames using IP addresses of our 25 Planet Lab servers as client-subnets, granting us a view of how each hostname resolves around the world. We begin with each set of responses grouped by the original hostname for which they are responses. We then cluster groups of responses which feature at least 10% overlap in A Records, in order to combine hostnames for hosts which belong to the same provider. While this may miss many of the configuration subtleties of CDN networks (*e.g.*, overlap in the networks, restrictions in IP-hostname mappings, *etc*), it allows us to paint a general picture of ECS enabled CDN sizes. This processing leaves us with a total of 383 providers.

Figure 4 shows a CDF of the number of IP addresses observed for each provider. First, we see the majority do not seem to feature large networks, with the median case having 5 IP addresses. Furthermore, nearly 90% of providers had fewer than 20 addresses. Despite the tendency towards small providers, we observed 40 providers with at least 20 IPs, and 17 with at least 50 IPs. Finally, we saw that 5 providers had more than 100 IPs, with the largest, belonging to Amazon, having over $3,000$. We emphasize that these values are certainly underestimates due to the small set of vantage points in our subnet pool. Indeed, using 25 IPs we observed 405 Google CDN IPs, while a previous study from 2013 discovered around 30,000 Google CDN IPs [17]. Still, our small-scale measurement grants us an understanding of the types of networks that are available to Fury Route.

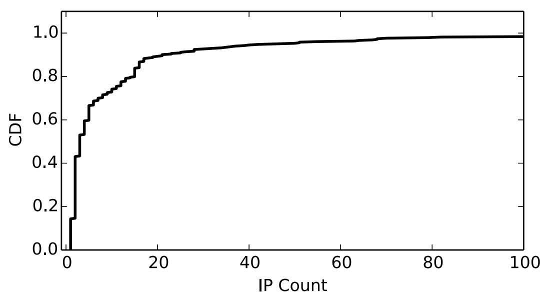

Figure 4: A CDF of the number of IPs observed from each provider cluster.

Provider	Hostname
Google	www.google.com
Edgecast	gp1.wac.v2cdn.net
Alibaba	img.alicdn.com
CloudFront	st.deviantart.net
CDN77	922977808.r.cdn77.net
CDNetworks	cdnw.cdnplanet.com.cdngc.net
ADNXS	ib.adnxs.com

Table 1: Selected set of providers.

2.4 Provider Granularity

We further examine the behavior of specific networks which are particularly useful in the development of Fury Route. We consider a set of CDN providers known to support EDNS [16, 17, 37] combined with our provider clusters from the above section. In order to avoid including the same provider twice that may have made it past our filtering process, we manually inspect any CNAME responses, as well as a sample of WHOIS records, ensuring that we select each provider once. While including multiple domain names from the same provider is not directly harmful, it complicates Fury Route's ability to make decisions, as we see in Section 4.

Table 1 shows our final selection of providers, as well as the corresponding hostname used to query each provider. We recall that providers are "used" by issuing a query for a hostname belonging to that provider and making use of the set of responding servers.

To understand the types of responses given by our selected providers, we consider the contents of their ECS

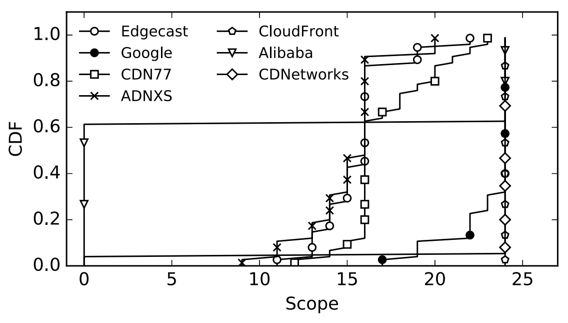

Figure 5: CDF of observed scope netmask responses.

responses. In particular, we examine the values of the response *scope netmask*, which indicates how well the subnet included in the query was able to match the response. While this value is likely affected by policy, *i.e.*, both the internal mapping policy of each provider, and a DNS caching policy which attempts to take advantage of DNS caching [20], the scope netmask is undoubtedly a valid asset in Fury Route's design. Fury Route will therefore interpret the values as the quality, *i.e.*, nearness, of a given response.

We query for each of our provider domain names with each of our 25 globally distributed IPs as the client subnet. Figure 5 shows a CDF of the response scopes for each provider. The providers fall into two categories: course and fine grained. CloudFront, CDNetworks, Google, return /24 (the smallest subnet recommended by the ECS specification) subnets for nearly all requests, appearing as vertical lines at 24 in the figure. Alternatively, Alibaba, ADNXS, and Edgecast return broader scopes. While CDN77 returns many broad scopes, we also see that nearly 40% of its responses were /18s or smaller. We note that some providers may employ anycast, suggesting any DNS client mapping they perform is intended for coarser grained locations.

The presence of varying granularities allows Fury Route to take advantage of multiple perspectives of the Internet. Indeed, if certain providers offer a coarse grained view, it will allow us to progress through the network differently than only fine grained providers which always offer very nearby results. Furthermore, using multiple providers enables us to gain multiple perspectives, as each network is managed and operates according to its own policy and deployments.

3. FURY ROUTE

Fury Route is built on the principle that the network distance between two hosts can be estimated by constructing a path of CDN replicas between the two hosts. These CDN replicas are returned as responses to ECS queries and the paths are generated by an iterative series of ECS queries which "hop" between CDN replicas by issuing new requests on behalf of a CDN replica with the client-subnet extension. This entire process can be performed from any host, requires no participation on the part of the hosts being measured, and does not rely on any directly deployed infrastructure. This is possible due to the nature of ECS, which allows any single probing node to issue DNS queries as if it were any other host. This allows Fury Route to perform its estimates with no direct probes, instead it only generates DNS queries to well provisioned DNS infrastructure.

The Fury Route system consists of three main components: *(i)* A chain building mechanism which connects an origin host with a destination host via a sequence of CDN replicas discovered via EDNS-enabled DNS responses, *(ii)* A voting system which enables this chain-building system to make forward progress in the space of CDN hosts, *(iii)* A comparison module, which compares the lengths of the chains and makes decisions about the relative distance between two points of interest, maximizing the information made available from the CDN-based DNS responses.

3.1 Chain Building

Fury Route is able to perform remote network distance estimations by using an approach we call *chain building*. The fundamental basis for this chain building approach is that DNS responses from CDNs which support ECS are likely to be *near* the requesting host, as indeed this is the stated purpose of ECS [20]. Fury Route builds on this notion, and constructs chains of near-by responses to estimate distance.

Fury Route begins with an origin host O and a destination host D. It further has a set of providers $P = \{p_1, \ldots, p_k\}$, where each p_i is represented by a hostname which belongs to a provider. While, as we saw in Section 2, a provider may span multiple hostnames, we take provider to mean an entity which can be queried by a look-up for a specific name. We therefore treat hostnames and providers as interchangeable.

To begin construction of a chain, Fury Route issues an ECS query to each provider in P, using D's address as the client subnet. It then takes the responses to each of these queries and pools them into a *target set*, which we denote $T = \{t_1, \ldots, t_n\}$, $n \geq k$.[1] These hosts represent CDN replicas which are likely close to the destination D, and therefore are indicators of its location. We use such a target set since the destination D may not be itself a CDN replica, but an arbitrary host. The target set therefore gives us a set of CDN replicas for which the algorithm is explicitly searching.

Next, Fury Route issues a set of ECS requests to the provider set P, using the origin host O's IP address as the client subnet. It then records the set of returned CDN replicas, noting their scope netmask values and

[1]The target set can contain more CDN replicas than the number of providers, because a provider may return more than a single replica for a host.

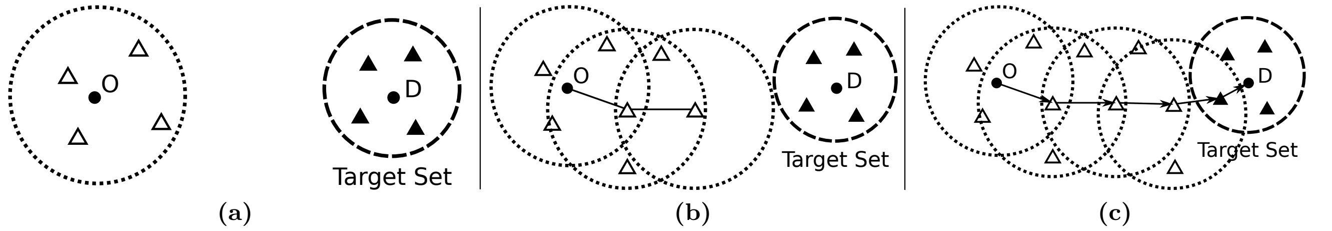

Figure 6: A representation of the chain building procedure. The dashed circle indicates hosts in the target set. Dotted circles show hosts in a scan of all providers.

the corresponding provider. It then considers each of such obtained CDN replicas, and selects one using the voting process described below. The voting procedure encourages the selection of hosts, *i.e.*, CDN replicas, which are closer to the target set T, and therefore closer to D. Fury Route then issues a new set of requests, using the previously selected CDN replica as its client subnet. This process is repeated until at least one provider returns a host which is in the target set T, or it exceeds a fixed number of scans. If it successfully encounters a replica from T, the resulting sequence of hosts is then taken as the *chain* of replicas connecting O and D.

Figure 6 shows a visual representation of these steps. Part (a) shows Fury Route's view after establishing the target set T (shown as shaded triangles within the dashed circle), and issuing the first set of ECS queries to the providers on behalf of the origin host. Non-shaded triangles show hosts returned as a result of those queries. Next, part (b) shows when it then selects one of these hosts, and issues another set of queries, offering a further set of CDN replicas. Finally, in part (c), the chain is complete, as the final round of queries to the providers returned results which land within the target set.

3.2 Voting

Fury Route employs a voting mechanism to select the next CDN replica host which is likely to provide forward-progress towards the destination host D. The voting mechanism is built on the heuristic that the best choice for the next hop is the one which brings the next hop closest to the target. To this end, we use the following mechanism: when considering a set of potential candidate CDN replicas, $C = \{c_1, \ldots, c_l\}$, Fury Route attempts to determine which will have the greatest overlap in ECS-enabled responses with the target set T.

In order to measure this overlap, Fury Route performs the following operation for each candidate CDN replica c_i. It issues an ECS query to the first provider, p_1, with c_i as the client subnet. We denote the set of responses as $R_{1,i}$. Next, it issues ECS queries to p_1 using each of the target CDN replicas in T. We combine all of the target set responses into a single collection which we denote $R_{1,T}$.

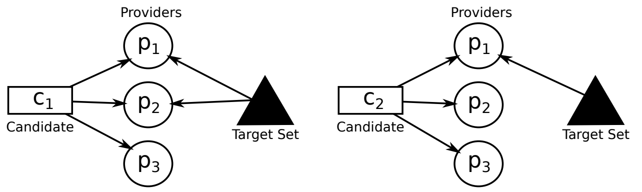

Figure 7: An example of the voting mechanism. Providers p_1 and p_2 have overlap with the target set for c_1, but only p_1 for c_2. Therefore the system selects c_1.

Using these sets we will determine which candidate is given the closest matching set of replicas to the target set. Formally, we measure the overlap seen by p_1 for candidate c_i, denoted $B_{1,i}$, as:

$$B_{1,i} = R_{1,i} \cap R_{1,T}.$$

If $B_{1,i}$ is non-empty, we say that this candidate has overlap with the target set as seen by provider p_1, and provider p_1 grants a single vote for c_i. If $B_{1,i}$ is empty, no vote is granted.

This process is repeated for each provider in P, and the votes are summed for the candidate. The entire process is further repeated for each potential candidate in C. It is important to note that a single provider may vote for many candidates. Fury Route then selects the candidate with the most votes, as it features the most overlap with the target set across providers, making it likely to offer the most forward progress.

Figure 7 presents a simple example of this process. In this example, we have 2 potential candidates, and 3 providers. We first query each of the providers and store the resulting responses. Next, each CDN replica host is scanned for each provider, also noting the results. In the example, we see that providers p_1 and p_2 see overlap with c_1, while only p_2 sees overlap with c_2. Therefore Fury Route selects c_1.

In the event that Fury Route finds itself with a set of candidates which have lower vote totals than the previous round, Fury Route "backtracks", abandoning the current chain branch, and returning to the candidate with the previous highest number of votes. If there is no such candidate, it then settles for the candidate with the next highest number of votes. In this way, it is able to pursue a path with the highest indication of progress, while avoiding moving further away from the target.

This process is aided by the variations in scope between the CDN providers. As we observed in Section 2.3, some providers use very specific locations, meaning overlap will only occur when the candidate is particularly close to the target set. Others use much coarser mappings, which allow overlaps, and therefore non-zero votes, to occur at greater distances. By combining a mix of such providers, this voting approach uses information from the coarse providers to initially move in the correct direction and depends increasingly on the finer grained providers as the chain of replicas approaches the target.

3.3 Chain Length

Once a chain has been constructed between the source and destination hosts as outlined above, the resulting chain is used as a comparison tool for estimating network distances. In particular, we take advantage of the response scope netmask field, which indicates the applicability of the answers. While this is largely intended for caching purposes, Fury Route makes use of the value to estimate the *quality* of a particular response, which we take to represent the accuracy of the chosen replica.

Fury Route therefore implements the following metric. If a chain includes a link between two CDN replicas in a chain, A and B, and s is the scope netmask of the response which included B, we take the cost of traversing such a link to be $\text{cost} = \max(8, 32 - s)$. The higher the precision of the response, the smaller the cost of the corresponding link. In a rare scenario when we obtain a scope s larger than 24 (*e.g.*, sometimes as large as 32), we "downgrade" such a response to 24, setting the minimal cost in the system to 8. Such adjustments prevent edge cases from creating chains with no information. Finally, responses which offer no scope information are ignored as being non-ECS supporting and therefore no such links will exist in a final chain.

4. IMPLEMENTATION

We now present the details of our implementation of Fury Route. Critical to its design is the lack of dependency on any external measurement infrastructure: no external vantage points are necessary, and neither the source nor the destination needs to be within the control of the measurement system. Therefore our implementation is able to run from a single machine.

In order to ensure flexibility when sending DNS requests, we use Google DNS and a modified version of the dnspython DNS library [8] to issue our queries. As in [37], we find that we are able to achieve up to 50 queries per second, depending on the provider set. All of our queries are sent with a full /32 client subnet. In the event that a chain fails to reach the target set after a threshold of candidate selection rounds, Fury Route abandons the current chain and starts over, attempting to build the chain from the destination to the origin.

4.1 Response Graph

In order to minimize the number of queries that must be performed for Fury Route to construct a chain, our implementation of Fury Route constructs a *response graph* while executing the chain construction. The response graph stores all observed replicas as nodes. Edges are used to encode the response scope from the perspective of different replicas. Nodes are further annotated with the set of providers which have been queried with that replica as a client subnet. Nodes within the same /24 subnet are combined, to avoid repeating queries with nearly identical client addresses.

This graph representation provides us with two key optimizations. First, it allows us to keep track of the queries sent and avoid sending duplicates, reducing the total query cost. Additionally, the added graph structure allows us to use replicas which may not occur in the chain itself. For example, the voting process may reveal additional replicas from candidate providers which were not explored directly during chain construction. Such paths may reveal "shortcuts" to the target set, eliminating potentially redundant components of the chain. After a single chain is completed, a shortest-path can then be taken from the source to the destination.

4.2 Provider Selection

The nature of providers is varied and complex, with each offering a potentially very different view of the Internet, as explored in Section 2.3. We restricted the current implementation to a relatively small set of providers in order to keep overhead at a manageable level. Indeed, adding too many providers may result in a significant increase in the number of candidates in each iteration of the chain-building process, which can introduce significant amounts of overhead. We explain how caching can help resolve such issues in the next section.

We divide our set of providers into two categories. The first, *voting-only* providers, are excluded from candidate selection in the chain building procedure because they lack sufficient accuracy. Nonetheless, they are very useful in voting due to their coarse-grained nature. The remaining providers are called *candidate* providers, which participate in both voting and chain construction. Based on our findings in Section 2, we take Alibaba, Edgecast, and ADNXS to be voting-only providers, due to their broader scope, *i.e.*, lower mapping accuracy. The remaining providers, CDNetworks, CloudFront, Google, and CDN77 are then taken as candidate providers.

To account for variations in the ECS scope netmask precision, we employ a corrective term when considering candidates from CDNetworks. In particular, we found that CDNetworks queries often returned servers that were quite distant from the node indicated in the client subnet, despite the high frequency of /24 scope responses. Fury Route therefore avoids taking such

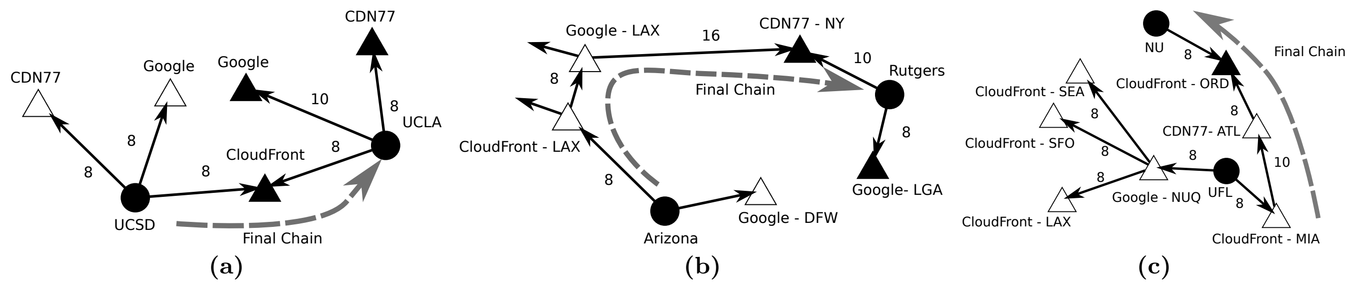

Figure 8: (a) A very near destination, (b) a chain which reflects the underlying CDN deployment, and (c) a chain which resulted from an exhaustive search.

routes, biasing against CDNetworks replicas by always setting them to a *single* vote. We further adjust for the underestimated scope by considering all corresponding edge weights at 3× the cost, a value which provides the best performance in our evaluations. The coarse nature of CDNetworks as candidates enables Fury Route to connect with distant locations, which might prove difficult with finer grained providers, *e.g.*, across oceans, as we explore later. The voting bias prevents more local paths from pursuing such routes unnecessarily.

4.3 Real-World Behaviors

We present several examples of the graph behavior exhibited by Fury Route. Separate from the full evaluation in Section 5, these descriptions aim to offer deeper understanding of the operational behavior of Fury Route.

Very-Near Destination In this case, we consider when the source is very near to the destination. In such situations, it is possible that several of the providers will direct both locations to the same replica.

Figure 8(a) demonstrates such a scenario between an origin server at UCSD and destination at UCLA. We see that on the first scan of the origin (UCSD in this case), a target node was encountered immediately. Since both origin and destination were quite near the replica, a short chain was promptly established.

Replica Sinks As Fury Route relies on the deployments the underlying CDNs, we expect that the chains constructed by Fury Route will likely reflect these deployments. One such example is the tendency of chains to include nodes which seem geographically out-of-the-way, but are reasonable given the CDNs networks, not unlike traditional packet paths and the underlying network. Our providers appear to have significant deployments in southern California, pulling many chains that direction, even if it is not the nearest geographically. To further understand the behaviors in this case, we examine the reverse DNS names of the replicas used by Fury Route. While not perfect identifiers, they provide a general idea of where a replica may be located [17].

Figure 8(b) shows an example of this when constructing a chain from an origin at Arizona to a destination at Rutgers. In this case, rather than passing through Texas, as a traceroute appears to do, the chain first passes through Los Angeles. It then makes a larger hop directly to New York, which brings it to the target set. While such paths may introduce some inaccuracy, Fury Route compensates by considering only *relative* chain lengths: as long as the origin is the same, the chains will encounter similar provider infrastructure layouts, and therefore encounter similar such detours.

Exhaustive Search Next, we consider a scenario in which the voting fails to immediately move Fury Route towards the best path. While it could still be driving the system in the correct direction, it may end up with a set of replicas where it can advance no further towards the destination. In these scenarios, Fury Route will explore the best voted paths, but will eventually abandon such branches and return to earlier options, via its backtracking mechanism. In these cases, Fury Route correctly determines the shortest path, despite its earlier explorations.

Figure 8(c) shows such a case between an origin at UFL and a destination at NU. Fury Route first explores the North American west coast before ultimately finding an appropriate chain to connect the two nodes. A Google response which sends Fury Route towards Mountain View, CA (NUQ), potentially as a fail-safe selection for Google, is not uncommon. We expect this behavior is the result of the underlying CDN deployment. Here, however, it is manifested in the voting system, rather than the candidate selection. Once none of the follow-up nodes in the west coast branch (CloudFront LAX, SFO, and SEA) have lower vote totals than the previous round, Fury Route abandons the west coast branch via the backtracking mechanism, and pursues the east coast path. In addition, since Fury Route takes the shortest completed path, it is able to avoid incurring any inaccuracy as the result of such scenarios.

Across Oceans As a final example (not shown in the figure), we consider a path which must cross an ocean. In particular, we consider an origin at UW in Seattle, and a destination at NITech in Nagoya, Japan. In this case, Fury Route pursues a number of candidates in Northern California (*e.g.*, Amazon-SFO and Google-NUQ), before selecting a CDNetworks candidate, with-

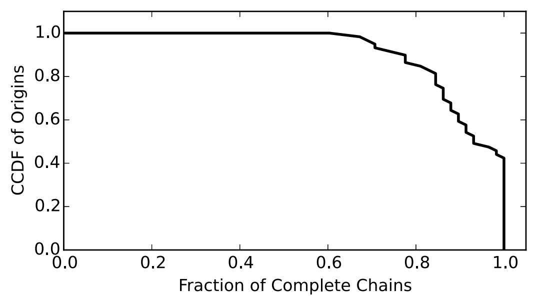

Figure 9: The fraction of destinations for which Fury Route was able to complete a chain.

out reverse DNS information. Finally, it then selects a target set replica from CDN77 in Tokyo.

Notably, CDNetworks replied to the request with a scope of /24. However, using this value directly would have resulted in a severely underweight chain, as the link from California to Japan introduces significant network distance. Therefore, our increased cost for CDNetworks nodes resulted in a more accurate final chain length, while still allowing Fury Route to successfully cross the gap caused by the Pacific ocean.

5. EVALUATION

In this section we explore the broader scale operation of Fury Route. We demonstrate that it is able to construct chains for a large fraction of globally distributed endpoints. We further evaluate its performance in estimating relative network distance. To understand which cases Fury Route performs poorly in, we examine the most prominently observed pathological behaviors. We then evaluate the potential of caching systems for Fury Route, and quantify the benefits they provide to a large scale system. Finally, we evaluate the effect that regularly shifting CDN replica selection has on Fury Route.

5.1 Completion Rate

First, we consider the completion rate of Fury Route chains. Here, we consider a set of 60 Planet Lab sites, distributed globally. While Fury Route enables network distance estimates between any two endpoints, we use Planet Lab nodes to obtain RTT measurements between them, *i.e.*, to establish the ground truth. For each site, we attempt to construct a chain to each of the other 59 sites. Each chain is given up to 25 candidate selection rounds in order to complete. If it exceeds this value, the chain is marked as incomplete. Experimenting with larger values provided no detectable increase in completion rate. Therefore, 25 provided a balance between exploratory freedom and run time.

Figure 9 shows the fraction of destinations for each origin server for which Fury Route was able to construct a chain as a CCDF over the set of origins. In the median case, 90% of chains are successfully completed, and in over 40% of cases, all chains were completed successfully. We found that pairs unable to complete their chains featured destinations with potentially sparse CDN deployments from our providers. Indeed the worst performing destinations were in Argentina (a particular destination saw 57% of chains failed), New Zealand (48%), and China (43%). However, even in cases in which the destination features a nearby replica, distant nodes may never be directed to such a node. For example, a CDN provider with a replica in South America may only ever send South American traffic to that replica, making it difficult for Fury Route to reach such destinations. As expected, locations that have poor CDN deployment, considering particular CDN providers and domain names that Fury Route uses, are most problematic both in terms of chain completion rate, and accuracy, as we demonstrate below.

5.2 Comparison To King

Here, we aim understand the performance of Fury Route compared to a technique with similar goals. In particular, we attempted to measure the performance of King [24] on the 60 Planet Lab hosts from our previous experiment. However, we found that King is no longer tenable: not a single one of our hosts had a name server that would respond to recursive queries.

To develop a further understanding of how King may perform on the modern Internet, we consider a random sample of 1000 addresses from IPv4 space. For each, we attempted to run King between the address and an address at our local institution. We again found very low rates of recursive resolvers in the wild, with only about 4% of addresses having name servers that would accept such queries.

While other mechanisms for estimating distance in the network exist, they all require significant infrastructure and measurements capabilities, as we discuss in Section 7. In cases when such infrustructure-dependent systems are not available, Fury Route may be the *only* source of information. We therefore consider its performance when compared to the ground truth of a direct latency measurement.

5.3 Rank Performance

Here, we evaluate Fury Route's ability to correctly estimate relative network distances. We show, given a fixed origin point, how well Fury Route is able to correctly determine which of a pair of destinations is closest and which is further away. Indeed, most real-world applications, including peer or server selection as well as Web performance and auditing, rely exclusively only on relative network distance. In order to test this, we again consider the full mesh of chains provided by our previous 60 Planet Lab nodes. To establish a ground truth network distance for each pair, we perform a ping measurement immediately prior to generating the Fury

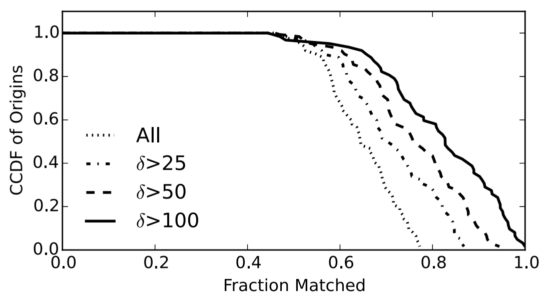

Figure 10: A CCDF over origin hosts showing the fraction of comparisons for which the Fury Route chain length matched the measured RTT ordering.

Route chain, granting an up-to-date view of the network delay between origin and destination.

With the above data, we wish to determine how well Fury Route's chain-lengths estimate the relative ordering given by the RTT measurements. We consider the set of all possible pairwise comparisons between destinations for each origin, giving us up to 1770 comparisons per origin (*i.e.*, 1,770 = 59 + 58 + ...), depending on each origin's completion rate. For each pair, we check if the comparison of the corresponding Fury Route chain lengths matches that of the corresponding ping measurements. We then are able to count the fraction of comparisons which matched the RTT measurements.

Figure 10 shows a CDF of the fraction of matched comparisons for each of our origins. The dotted line to the left indicates all possible pairs. Each other line indicates the performance for the subset of comparisons (among the 1770 pairs) with a minimum distance between path RTTs of δ. For example, for an origin S, $\delta = 25$ contains all pairs of destinations, *e.g.*, A and B, for which $|\mathrm{RTT}(S,A) - \mathrm{RTT}(S,B)| > 25ms$. For our set of hosts, 71% of pairs were in $\delta > 25$, 54% in $\delta > 50$, and 29% in $\delta > 100$. First, we note that in nearly all cases, Fury Route is above 50% performance in terms of matches. Furthermore, we see that increasing the difference between the origin and destinations expectedly improves performance. Indeed, the best case of a difference of $\delta = 100$ gives us 83% of comparisons correct in the median case.

Such results are in line with our expectations: since Fury Route is not measuring network delay values directly, we do not expect it to be able to differentiate small differences in round trip time. Moreover, such cases are not likely to have significant effects on network performance. Furthermore, Fury Route is able to provide the valuable network distance information about two remote hosts that may otherwise be inaccessible.

5.4 Pathological Cases

Next, we present the notable pathological behaviors encountered by Fury Route during the course of our evaluation. In general, these cases violate the assumptions on which Fury Route operates, causing inaccuracies and other performance anomalies.

No Nearby Replicas In this scenario, an origin or destinations may be distant from all CDN replicas served to it. As discussed in Section 5.1, such situations may prevent Fury Route from constructing a chain. However, in the event that they do connect, they are likely to provide highly inaccurate paths, as Fury Route may severely underestimate the distance to such end points. For example, a destination host at SJTU in Shanghai frequently returned CloudFront nodes in Los Angeles with a scope of /24. This ultimately resulted in a large number of underestimated distances. Similar behavior was observed for a destination in Buenos Aires returning responses in Indiana (CloudFront) and Atlanta (CDN77). Indeed, the lack of CDN deployment, CloudFront in Asia and South America in this case, is taking its toll.

Missed Replicas In this scenario, there is a clear link which Fury Route misses because it was unable to recognize the proximity of two replicas. The precise reason for this is likely the result of internal CDN policy. As an example, an attempted chain from UFL to Rutgers featured a CloudFront replica in the target set with a reverse DNS name suggesting proximity to Washington D. C. (*i.e.*, IAD). During chain construction, Fury Route regularly encountered other CloudFront nodes also named as being near Washington D. C., but from a significantly different subnets. Fury Route was unable to recognize their proximity, and therefore generated a significantly longer path, ultimately overestimating the network distance.

Bad Hops A final observed pathological behavior occurs in the interactions between providers. Specifically, a provider may respond with a close match (*i.e.*, a /24), but return addresses for clearly very distant servers. For example, occasionally when given a client subnet from CDN77 in NYC, Google will respond with a node in Paris, as observed by reverse DNS name. While such a jump does not necessarily hurt quality of Fury Route's final chain, it increases the likelihood that the chain will include a significant detour that is not evenly reflected throughout all chains.

While these pathologies result from several observed behaviors, they have two core consistent features. First, they arise from variation in CDN policy or behavior, in either the specific value of the ECS response scope, or in the particular replica selection policy employed by the CDN. Second, they are rooted in the lack of full global deployment of the some of selected CDNs. Consequently, we expect their occurrence will decrease as the density of the CDN providers increases. Moreover, utilizing semantic CDN information, and applying learning and blacklisting methods could help purify the Fury Route response graphs.

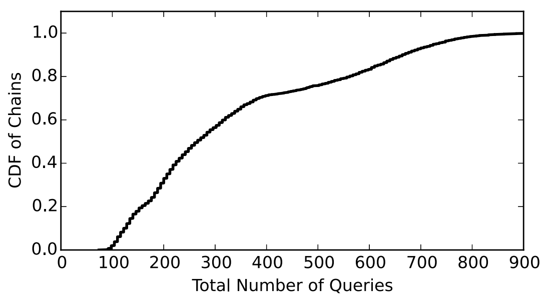

Figure 11: CDF of the queries used for constructing each chain.

5.5 Overhead Analysis

Next, we measure the cost of Fury Route in terms of the DNS query overhead. In particular, we explore the number of queries required to successfully construct a chain. We reiterate that the network-level overhead is entirely based on DNS queries: Fury Route generates no direct network measurements and therefore generates no load on the hosts being measured.

Figure 11 presents a CDF of the number of queries to complete all the chains in the previous section. In the median case, a single chain requires 260 queries, which can be completed in under 6 seconds. Furthermore, nearly 90% of chains are able to complete with under 650 queries. The increase in queries beyond 400 is the result of destinations which had to invoke the reverse path in order to complete, and therefore had to construct a second chain. Since our providers are all large-scale networks, chain construction does not represent significant load on their networks.

Graph Caching The use of our graph data structure to implement Fury Route comes with the additional benefit of creating a simple and effective caching mechanism. If a large number of measurements are being performed from a single origin, as is the expected use-case, Fury Route can simply reuse the graph for each subsequent chain construction. With a sufficient cache, queries could then be executed extremely rapidly, making Fury Route viable as a real time estimation tool.

To quantify the benefits of graph caching, we conduct the following experiment. First, we randomly sample 50 Planet Lab nodes as origins. For each origin, we randomly sample 200 addresses from IPv4 space. Next, we construct chains from each origin to each of its corresponding destinations, reusing the graph for each origin.

Figure 12 presents the average number of queries for each origin: the x-axis indicates the query index, *i.e.*, how many times the graph has been reused, and the y-axis is the average number of queries, where the error bars represent a standard deviation. We see that the initial chain takes an average of 250 queries to complete, but quickly decreases, requiring only 65 queries by the 10th chain constructed with the graph. After 20 chains are constructed, the average number of queries

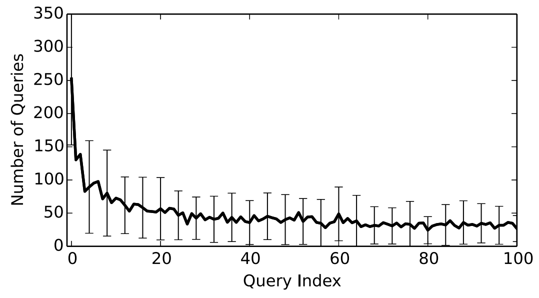

Figure 12: The improvement of queries needed over time with the same graph.

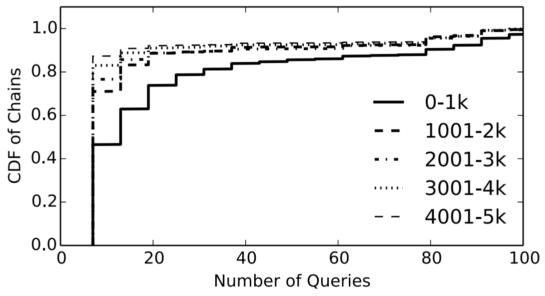

Figure 13: A CDF of the queries needed for each set of 1000 chains.

decreases below 50. The jittery nature of this data is due to the fact that occasionally one of the randomly generated IPs will be a significant distance away, requiring exploration of replicas not previously seen.

To further explore development of the graph cache, we conduct an experiment in which we use a single origin and graph to construct chains to 5000 randomly sampled IP addresses. In doing so, we seek to understand the fraction of chains which can be constructed entirely from the contents of the cache.

Figure 13 shows a CDF of the number of queries needed for each set of 1000 chains. We see that the initial 1000 bears most of the query-burden, and a single chain takes up to 250 queries to complete. However, we see that by the end of these first 1000 chains, 47% of queries needed only the minimum 7 total queries to complete, *i.e.*, they completed after only scanning the destination for the 7 domain names shown in Table 1, the smallest overhead possible. As more chains are constructed, we observe a decrease in the number of queries required, until after 4000 queries, 87% of chains are able to complete with minimum overhead.

This graph caching principle could be extended to a complete cache, reducing all chains to the minimum number of queries. Previous work [17] has demonstrated that the entire /24 client space can be explored for Google in approximately a day using a single resolver. Extrapolating this to our other providers, one could pay the one time cost of scanning each provider to discover their entire current set of replicas and the corresponding

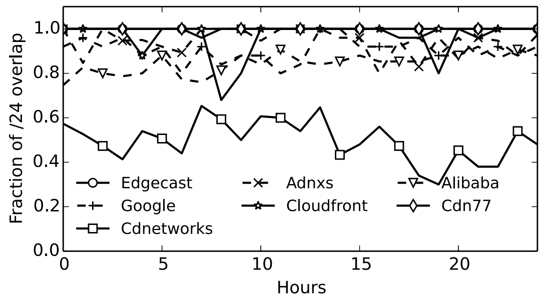

Figure 14: Similarity in DNS responses for each provider over a day.

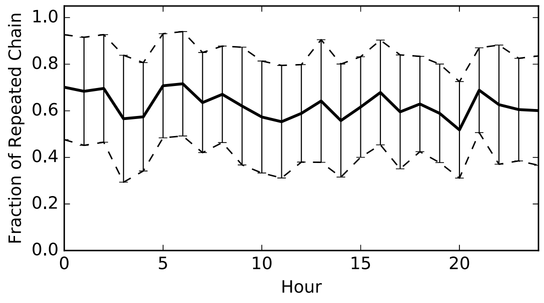

Figure 15: The average fraction of replicas in chains which remained the same over time

edges in the Fury Route graph. Still, our experiment in Figure 13 demonstrated that an organically constructed graph can be built in under 2 hours which provides near instant measurements for a large fraction of queries.

5.6 Change Over Time

By design, CDN responses to DNS queries can be highly variable, incorporating client location, internal cache state, load balancing policy, and any other provider decision making policy. We therefore examine the consistency of Fury Route's performance over time. We examine how responses change over time, and the effects this has on the resulting chains. To quantify the underlying variation in CDN replica selection, we queried each of our providers with our 25 Planet Lab nodes from Section 2 every hour for a 24 hour period.

Figure 14 shows the fraction of /24 subnets that were observed in the previous hour's response. For the majority of providers, 80% of the provided subnets are the same, suggesting moderate replica churn. In the case of CDN77, we observe no variation for the entire 24 hours. An exceptional case can be found with CDNetworks, which averaged only a 50% match. Further analysis revealed this was partially due the the fact that the majority of responses from CDNetworks featured only 2 A records, contrasted with Google's regular 5 or more.

Next, we examine the effect the variation in provider responses has on Fury Route's chains. We select 30 random origin-destination pairs from our set of Planet Lab nodes and generate each chain between them every hour for 24 hours. Figure 15 shows the average fraction of replicas in a chain that appeared in the chain for the previous hour, *i.e*, the fraction of overlap with the previous chain, where error bars indicate a standard deviation. At each sample, the overlap varies between 60 and 75%. The relatively large standard deviation further indicates significant changes for some chains.

To understand the effect of variation on Fury Route's performance, we consider the following experiment. We again generate the 60^2 mesh of chains, this time allowing each origin to construct a graph cache. After 12 hours, we construct the chains again, re-scanning all nodes and combining the cached graph with nodes revealed in the fresh scan. In the second round of chain building, each graph grew an average of 26%, as a significant set of new replicas were observed, in line with Figure 15. However, the results show that the two sets of chains presented *no* change in rank performance. This shows that Fury Route is able to generate meaningful representations of the network with a single snapshot, and there is no need to capture all time variations.

6. DISCUSSION

Non-Client Mapped Responses As we demonstrated in Section 2, not all providers who support ECS use it to perform replica selection. Even in cases where the provider performs client mapping for some responses, it does not always do so (*e.g.*, Alibaba). Such cases are necessarily unhelpful to the operation of Fury Route. However, as long as the non-ECS nature is indicated, *i.e.*, a 0 response scope, Fury Route will ignore it entirely. In the event that it has a nonzero scope, Fury Route may be able to work around such responses, as such candidates will ultimately receive few votes during chain construction.

ECS Adoption The behavior of Fury Route is likely to change as ECS adoption increases. Indeed, as the number of visible replicas from which Fury Route can construct chains increases, the quality of the chains for path distance estimation is likely to increase. Furthermore, additional votes in our chain construction mechanism will only increase the accuracy of decisions that the system makes.

Additional Providers As discussed previously, the current implementation of Fury Route relies on a set of detected, measured, and processed providers who support the EDNS0 client-subnet extension. However, a larger number of providers could be selected, in particular, as the density of networks increases a greater number of providers becomes available. However, doing so comes at the cost of increasing the number of queries performed by Fury Route: there are a greater number of candidates and a greater number of voters at each iteration.

7. RELATED WORK

There has been a significant body of work devoted to the challenge of predicting network performance. These have included large-scale measurement platforms [33, 34, 35, 27, 28], which attempt to measure a large number of routes and hosts from a large number of vantage points. Other systems have embedded coordinate systems, often based on measurements to a set of known landmarks or peers, to perform network distance estimations between a set of hosts without direct measurements [21, 22, 23, 30, 36, 40]. Contrary to the above systems, Fury Route outsources the direct network measurements to a number of underlying CDNs. As a result, it conducts neither passive nor active measurements of any sort. It further requires no access to the measured endpoints nor to any other third-party hosts or infrastructure.

King examines how latency can be measured indirectly by considering the latency between two nearby DNS resolvers [24]. While similar to Fury Route in that it does not require the direct participation of either host, King requires a nearby open recursive resolver, and a nearby authoritative server. However, such requirements are becoming more difficult to satisfy. A recent study has shown that the number of open recursive DNS servers is rapidly decreasing – approximately by up to 60% a year, and by around 30% on average a year [26]. In addition, the DNS hosting is another significant and enlarging problem for King, as discussed in Section 1. Moreover, our experiments found that only 4% of a random sample of Internet hosts had open recursive DNS servers, rendering King widely unusable. While Fury Route utilizes the DNS infrastructure, it is in no way dependent on the number or availability of open DNS resolvers, nor is it affected by DNS hosting. Fury Route requires only that a host has nearby CDN replicas, a situation CDNs are strongly motivated to maximize.

The use of CDN redirections has been shown effective in terms of relative network positioning [38, 39]. In particular, if two clients have overlapping CDN replicas, they are likely to be close to each other in the network sense. Such an approach has further been utilized by large-scale systems such as BitTorrent [19]. Contrary to such an approach, which requires a large-scale distributed system such as BitTorrent in order to be effective (so that two BitTorrent clients have an overlap among CDN replicas), Fury Route has no such limitation. Indeed, it can, in principal, effectively connect any two endpoints on the Internet. The key behind this is the use of multiple CDNs of different granularity in terms of CDN replicas, *i.e.*, both fine- and coarse-grained, and the use of ECS as a vehicle to hop over different CDN replicas.

Further work has been done to develop deeper understanding of the behavior of networks in Reverse Traceroute [25]. Reverse Traceroute combines a number of techniques to piece together a reverse path from a host. While Reverse Traceroute has a fundamentally different goal from Fury Route, it is similar in its aims of allowing measurement to and from an arbitrary host. However, Reverse Traceroute relies on a number of pieces of participating infrastructure to collect trace route probe information. It further requires access to the source node. While Fury Route is built on the back of the CDNs' infrastructure, it relies on these CDN networks to perform the "heavy lifting," and needs only ECS queries. Furthermore, contrary to Reverse Traceroute, it requires no access to any of the two measuring endpoints and no participating measurement infrastructure.

Finally, the use of ECS [20] as a measurement tool was the key principle in [17, 37]. While similar in our use of ECS to obtain client-mapping information from existing infrastructure, both of these works have a different goal: exploring the deployments of specific CDNs. Fury Route, on the other hand, is attempting to use these CDNs to perform an additional task: network distance estimation. Therefore, while we build on the complexity and heterogeneity of the CDNs measured in these works, Fury Route solves an entirely different problem with ECS as a tool.

8. CONCLUSIONS

In this paper, we presented Fury Route, a system which builds on the underlying client mapping performed by CDNs and the potentials of the EDNS client subnet extension. Fury Route constructs chains of responses and uses the lengths of these responses to estimate the relative network distance between remote hosts, all without any direct network measurements. We presented a study of a number of behaviors observed in Fury Route when constructing response chains in the real world. We demonstrated Fury Route's ability to construct chains to 90% of destinations in the median case. We further showed that it is successfully able to select a best-choice, choosing correctly in up to 83% of median cases. Additionally, we examined the potential for caching, showing a significant capability for caching route graphs, allowing over 87% of chains to complete with the minimum possible number of queries after a cache has been built and demonstrated the longevity of such caches. Given its lack of requirement for directly controlled measurement infrastructure, low overhead, and ability to measure between arbitrary hosts, Fury Route stands to be a practical and powerful tool for estimating relative network distance.

9. REFERENCES

[1] Akamai: Fast DNS. `https://www.akamai.com/us/en/solutions/products/cloud-security/fast-dns.jsp`.

[2] Amazon Route 53. `https://aws.amazon.com/route53/`.

[3] Azure DNS. `https://azure.microsoft.com/en-us/services/dns/`.

[4] CDNetworks Cloud DNS. `https://www.cdnetworks.com/products/cloud-dns/`.

[5] Cloudfare: Global, fast and always secure authoritative DNS. `https://www.cloudflare.com/dns/`.

[6] DNSMadeEasy. `http://www.dnsmadeeasy.com/`.

[7] Dyn DNS. `http://dyn.com/dns/`.

[8] ECS for dnspython. `https://github.com/mutax/dnspython-clientsubnetoption`.

[9] GoDaddy: Premium DNS. `https://www.godaddy.com/domains/dns-hosting.aspx`.

[10] Google Cloud Platform: Cloud DNS. `https://cloud.google.com/dns/`.

[11] Google Public DNS. `https://developers.google.com/speed/public-dns/docs/using?hl=en`.

[12] Neustar DNS Services. `https://www.neustar.biz/services/dns-services`.

[13] OpenDNS. `https://www.opendns.com/`.

[14] Verisign Managed DNS. `http://www.verisign.com/en_US/security-services/dns-management/index.xhtml`.

[15] Verizon ROUTE: Fast, Reliable Enterprise-Class Services for Domain Name System (DNS). `https://www.verizondigitalmedia.com/platform/route/`.

[16] Which CDNS support EDNS-client-subnet. `https://cloud.google.com/dns/`.

[17] Calder, M., Fan, X., Hu, Z., Katz-Bassett, E., Heidemann, J., and Govindan, R. Mapping the expansion of Google's serving infrastructure. In *Proceedings of IMC '13* (2013), IMC '13.

[18] Chen, F., Sitaraman, R., and Torres, M. End-user mapping: Next generation request routing for content delivery. In *Proceedings of ACM SIGCOMM '15* (London, UK, Aug. 2015).

[19] Choffnes, D., and Bustamante, F. Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *Proceedings of ACM SIGCOMM '08* (Seattle, WA, Aug. 2008).

[20] Contavalli, C., van der Gaast, W., tale, and Kumari, W. Client subnet in DNS queries(IETF draft). `http://www.ietf.org/internet-drafts/draft-ietf-dnsop-edns-client-subnet-06.txt`.

[21] Costa, M., Castro, M., Rowstron, A., and Key, P. PIC: practical Internet coordinates for distance estimation. In *Proc of ICDCS '04* (2004).

[22] Dabek, F., Cox, R., Kaashoek, F., and Morris, R. Vivaldi: A decentralized network coordinate system. In *Proc. of SIGCOMM '04* (2004), SIGCOMM '04.

[23] Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. IDMaps: A global Internet host distance estimation service. *IEEE/ACM Trans. Netw. 9*, 5 (Oct. 2001).

[24] Gummadi, K., Saroiu, S., and Gribble, S. King: Estimating latency between arbitrary Internet end hosts. In *Proceedings of Internet Measurement Workshop (IMW)* (Marseille, France, Nov. 2002).

[25] Katz-bassett, E., Madhyastha, H. V., Kumar, V., Scott, A. C., Sherry, J., Wesep, P., Anderson, T., and Krishnamurthy, A. Reverse traceroute. In *Proc of. USENIX NSDI '10* (2010), NSDI '02.

[26] Kuhrer, M., Hupperich, T., Bushart, J., Rossow, C., and Holz, T. Going wild: Large-scale classification of open DNS resolvers. In *Proceedings of IMC '15* (2015), IMC '15.

[27] Madhyastha, H. V., Anderson, T., Krishnamurthy, A., Spring, N., and Venkataramani, A. A structural approach to latency prediction. In *Proc. of ACM IMC '06* (2006), IMC '06.

[28] Madhyastha, H. V., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., and Venkataramani, A. iPlane: An information plane for distributed services. In *Proc. of OSDI '06* (2006), OSDI '06.

[29] Narayanan, A., Nam, Y., Sivakumar, A., Chandrasekaran, B., Maggs, B., and Rao, S. Reducing latency through page-aware management of web objects by content delivery networks. In *Proc. of ACM SIGMETRICS '16* (2016).

[30] Ng, T., and Zhang, H. Predicting Internet network distance with coordinates-based approaches. In *Proc. of IEEE Infocom '02* (2002).

[31] Otto, J. S., Sánchez, M. A., Rula, J. P., and Bustamante, F. E. Content delivery and the natural evolution of DNS: Remote DNS trends, performance issues and alternative solutions. In *Proc. of ACM IMC '12* (2012), IMC '12.

[32] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings*

of ACM SIGCOMM '98 (Vancouver, British Columbia, Sept. 1998).

[33] Rabinovich, M., Triukose, S., Wen, Z., and Wang, L. DipZoom: The Internet measurements marketplace. In *Proc of IEEE Infocom '06* (2006).

[34] RIPE Atlas. `https://atlas.ripe.net/`.

[35] Sánchez, M. A., Otto, J. S., Bischof, Z. S., Choffnes, D. R., Bustamante, F. E., Krishnamurthy, B., and Willinger, W. Dasu: Pushing experiments to the Internet's edge. In *Proc. of USENIX NSDI* (2013).

[36] Shavitt, Y., and Tankel, T. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *Proc. of INFOCOM '04* (2004).

[37] Streibelt, F., Böttger, J., Chatzis, N., Smaragdakis, G., and Feldmann, A. Exploring EDNS-client-subnet adopters in your free time. In *Proceedings of IMC '13* (2013), IMC '13.

[38] Su, A.-J., Choffnes, D., Bustamante, F., and Kuzmanovic, A. Relative network positioning via CDN redirections. In *Proceedings of ICDCS '08* (2008), ICDCS '08.

[39] Su, A.-J., Choffnes, D., Kuzmanovic, A., and Bustamante, F. Drafting behind Akamai (Travelocity-based detouring). In *Proceedings of ACM SIGCOMM '06* (Pisa, Italy, Sept. 2006).

[40] Wong, B., Slivkins, A., and Sirer, E. G. Meridian: A lightweight network location service without virtual coordinates. In *Proc of SIGCOMM '05* (2005), SIGCOMM '05.