# Fury Route: Leveraging CDNs to Remotely Measure Network Distance

Marcel Flores, Alexander Wenzel, Kevin Chen, and 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 [16]. Such estimates have been shown to heavily correlate with actual end-to-end performance (in terms of throughput and delay) between the two endpoints [23, 30].

With King [16], 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 [17]. On the other hand, DNS hosting (*i.e.*, outsourcing DNS services to the cloud [1–3, 5–8]), is fundamentally

blurring the assumption of co-location of endpoints (both clients and servers) and authoritative DNS servers.

We present Fury Route, a system that aims to estimate the network distance between arbitrary Internet endpoints. Fury Route relies on (*i*) the existence of different Content Distribution Networks (CDNs) and their heterogeneous deployment (*ii*) CDNs' common desire to direct clients to nearby CDN replicas, and (*iii*) the proliferation of EDNS0 client-subnet extension (ECS) [12], a mechanism by which a host issuing DNS requests can indicate the origin of the request *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.* 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 by sending or responding to probes. Fury Route generates no load on the parties involved: It only queries DNS, whose infrastructure is designed to handle large loads. While Fury Route utilizes the DNS infrastructure, it is in no way impacted by availability of recursive DNS resolvers, nor is it affected by DNS hosting. Fury Route utilizes the mapping work done by CDNs, and it effectively extracts this information via DNS.

We evaluate Fury Route using ground truth obtained from PlanetLab and RIPE Atlas platforms, testing from around 9000 nodes well distributed across countries and networks. We find that in the median case, Fury Route is able to construct chains between more than 80% of origin and destination pairs. This significantly outperforms other evaluated systems, *i.e.*, King [16] and iPlane [19], which have the convergence rate of 4% and 56% on the same data set, respectively. We further demonstrate that despite its infrastructureless properties, Fury Route shows accuracy comparable to iPlane, which conducts large-scale Internet measurements for this purpose. In particular, Fury Route is able to correctly order up to 83% of destinations in the median case. We further show our graph caching technique is able to reduce queries by 80%.

## 2 Background and Measurement

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 [12, 22]. 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.

### 2.1 Provider Granularity

We 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 [9, 10, 28] combined with a set of providers collected via manual inspection from a scrape of the Alexa Top500 [1].

| 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.

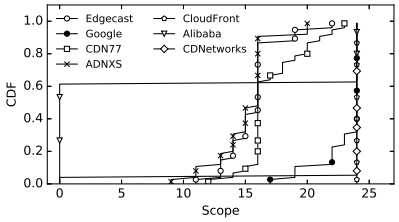


**Fig. 1:** CDF of observed scope netmask responses.

Table 1 shows our selection of providers and the corresponding hostname used to query each provider. Providers are "used" by issuing an A record query for a hostname belonging to that provider. We examine the response *scope netmask*, which indicates how well the subnet 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 [12], 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 each of the provider domain names using 25 distinct globally distributed prefixes from PlanetLab as the client subnet. Figure 1 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 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.

[1] Akamai, a large provider, restricts third-party ECS queries, and is not used.

## 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. The intuition is that the replicas provide a reflection of the density of CDN deployments: crossing low density areas suggests large distances.

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 as ECS allow single probing node to issue DNS queries as if it were any other host. While these responses may vary due to outside factors, in particular when examining the precision of an ECS response which used a CDN replica as a client subnet (a case for which they are unlikely to be optimized), they still contain information which reflects the structure of the underlying networks. Fury Route addresses this by using the subnet mask returned by the ECS query: poor matches usually come with generic answers and large subnets, which translate to large distances.

Fury Route 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 estimates the relative distance between two points of interest, maximizing the information made available from the CDN-based DNS responses.

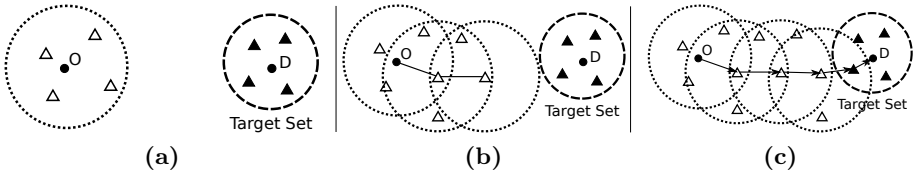


**Fig. 2:** A representation of the chain building procedure. The dashed circle indicates hosts in the target set. Dotted circles show hosts in a scan of all providers.

**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 [12]. 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 $T = \{t_1, \ldots, t_n\}$, $n \geq k$.[2] 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 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 2 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.

**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 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 $R_{1,T}$.

[2] 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.

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, choosing randomly in the case of ties.

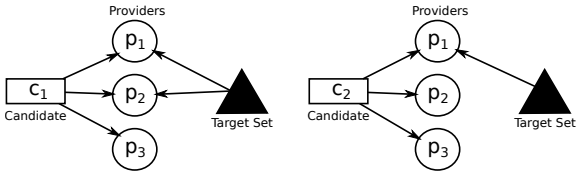


**Fig. 3:** An example of the voting mechanism. Providers $p_1$ and $p_2$ have overlap with the target set for $c_1$, but only $p_1$ for $c_2$. Therefore the system selects $c_1$.

Figure 3 presents an 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 responses. Next, each CDN replica host is scanned for each provider, also noting the results. In the example, 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.

**Chain Length** Once a chain has been constructed between the source and destination hosts as outlined above, the *length* is used as a relative comparison tool against other chains for estimating network distances. To compute the length, we use the response scope netmask field. 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. 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 cost $= \max(8, 32 - s)$. The higher the precision of the response, the smaller the cost of the corresponding link. In rare cases, we

obtain a scope larger than the ECS specification's maximum length recommendation of 24. These responses are inconsistent and do not occur reliably across our providers. We "downgrade" such responses to 24, setting the minimal cost in the system to 8. Finally, responses which offer no scope information are ignored.

## 4 Implementation

Queries are issued to Google DNS with a modified version of the dnspython DNS library [4] to issue our queries. As in [28], we are able to achieve up to 50 queries per second, depending on the providers. All of our queries are sent with a full /32 client subnet. If 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.

Fury Route builds a *response graph* to minimize the number of queries it performs. 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.

### 4.1 Provider Selection

We divide our set of providers into two categories. The first, *voting-only* providers, are excluded from candidate selection in the chain building procedure as they lack sufficient accuracy. Nonetheless, they are very useful in voting due to their coarse-grained nature. The second, 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. The remaining CDNs, are taken as candidate providers.

**"Unmapped" Replica Blacklisting** In examining pathological cases, we observe that they arise from variation in CDN policy or behavior in either ECS response scope or replica selection policy. A common signature for a "poor" CDN decision is that in most cases, in absence of an informed response, CDNs practice directing such queries to "unmapped" CDN replicas. Such CDN replicas are typically recommended when a request is conducted from an address with no suitable mapping. The "unmapped" CDN replicas are easy to detect in the context of Fury Route, since the number of such CDN responses typically outweigh a regular CDN replica by up to an order of magnitude. We demonstrate that avoiding chains through such "unmapped" replicas enables our system to retain high path completion rate while avoiding extremely erronous results.

## 5 Evaluation

In this section we evaluate Fury Rotue with two different platforms that provide ground truth round-trip time measurements. First, we use 8,964 globally-distributed nodes from a publicly available RIPE Atlas platform [25]. Second,

we consider a full mesh of chains provided by a set of 60 globally-distributed Planet Lab nodes. To establish a ground truth network distance for each pair, we perform a ping measurement (consisting of three pings) immediately prior to generating the Fury Route chain, granting an up-to-date view of the network delay between origin and destination.

### 5.1 Completion Rate

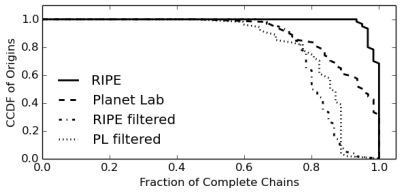


**Fig. 4:** The fraction of destinations for which Fury Route was able to complete a chain in different scenarios.

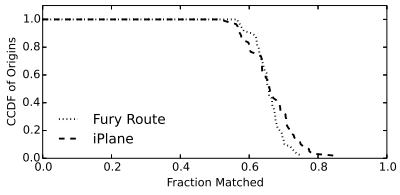


**Fig. 5:** The ranked ordering performance of Fury Route and iPlane on RIPE Atlas for completed chains.

First, we consider the completion rate of Fury Route chains in each platform. For each pair of tested nodes, we attempt to construct a Fury Route chain. Each chain is given 25 candidate selection rounds before it is marked incomplete. Larger values provided no detectable increase in completion rate, so 25 provided a balance between exploratory freedom and run time.

Figure 4 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. First, we focus on "raw" results, marked as "RIPE" and "Planet Lab." We see that in the median case for Planet Lab, 90% of chains are successfully completed, and in over 40% of cases, all chains were completed successfully. The results are even better for the RIPE data set, where 100% of chains are successfully completed in the median case. We found that pairs unable to complete their chains featured destinations with potentially sparse CDN deployments from our providers.

The other two curves, "RIPE filtered" and "PL filtered," show the chain completion rates for the filtered scenarios, *i.e.*, when "unmapped" replica blacklisting, explained in Section 4.1 above, was applied. Such filtering decreases the completion rate, such that it becomes almost identical for the two platforms. Here, most of the "bad" paths, particularly in the RIPE data set, which completed but were of poor quality, are filtered in this step. Nonetheless, Figure 4 shows that the median Fury Route chain completion rate remains above 80%.

### 5.2 Comparison to iPlane

Here, we compare Fury Route's performance to the performance of an infrastructure-dependent system, iPlane [19] using RIPE Atlas. iPlane is a system for analyzing and predicting Internet path performance. It uses a distributed infrastructure

to compile traceroutes from various vantage points in order to predict network paths and path attributes [19]. We show, given a fixed origin point, how well Fury Route and iPlane are able to correctly determine which of a pair of destinations is closest and which is further away. For each pair, we check if the comparison of the corresponding Fury Route chain lengths and iPlane's latency estimates matches that of the corresponding ping measurements. We then are able to count the fraction of comparisons which matched the RTT measurements.

We first compare the completion rates. For Fury Route, the curve "RIPE filtered" in Figure 4 shows the median completion rate is approximately 80%. Our evaluation of iPlane shows a completion rate of 56% for the "RIPE filtered" set. Next, Figure 5 shows the performance of Fury Route and iPlane ranking RIPE Atlas when completion is possible. The curves show a CDF of the fraction of matched comparisons for each of our origins for all possible pairs. The performance of Fury Route and iPlane ranking is virtually identical in the median case. Fury Route achieves comparable performance to iPlane using only DNS and CDN deployment properties instead of iPlane's necessary back-end measurement network, while significantly outpeforming iPlane.

### 5.3 Rank Performance

Here, we analyze Fury Route's performance on a different, Planet Lab based, platform. We wish to determine how well Fury Route's chain-lengths estimate the relative ordering given by the RTT measurements in a 60-node full mesh scenario. We consider 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 completion rate.

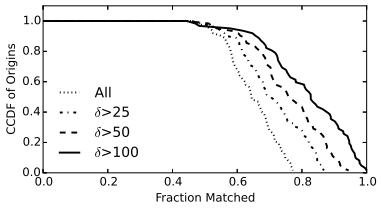


**Fig. 6:** A CCDF over hosts showing the fraction of comparisons the chain length matched the measured RTT ordering.

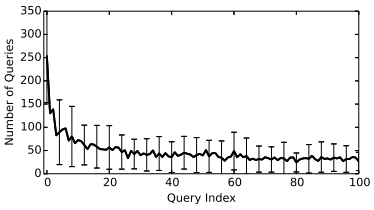


**Fig. 7:** The improvement of queries needed over time with the same graph.

Figure 6 shows a CDF of the fraction of matched comparisons for each of our origins. The dotted line to the left indicates all possible pairs. This result is similar and corresponds to the one shown above in Figure 5. Each other line indicates the performance for the subset of comparisons with a minimum distance between path RTTs of $\delta$. 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 $\delta > 100$. 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.

Many of Fury Route's misestimates stem from limits in the underlying CDN infrastructure. Targets in areas with limited deployments result in greater error in the initial hops, degrading the estimates. For example, such behavior was observed in Africa, South America, or when crossing oceans. Figure 6 shows the clearest tradeoff: when comparing similar distances, Fury Route becomes less accurate, as noise begins to dominate.

### 5.4 Overhead Analysis

The use of a graph to implement Fury Route provides a simple and effective caching mechanism. Fury Route can reuse the graph for multiple measurements from a single origin (the expected use case). With a sufficient cache, queries could then be executed in seconds, 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 7 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 decreases below 50.

## 6 Related Work

A significant body of work has been devoted to the challenge of predicting network performance. These have included large-scale measurement platforms [18, 19, 24–26], 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 [13–15, 21, 27, 31] Unlike these, Fury Route outsources the direct network measurements to a number of underlying CDNs. As a result it requires no access to the measured endpoints nor to any other third-party infrastructure.

King examines how latency can be measured indirectly by considering the latency between two nearby DNS resolvers [16]. 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 [17].

The use of CDN redirections has been shown effective in terms of relative network positioning [20, 29, 30]. 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 [11]. Contrary to such an approach, which requires a large-scale distributed system such as BitTorrent in order to be effective Fury Route has no such limitation. Indeed, it can, in principal, effectively connect any two endpoints on the Internet.

Finally, the use of ECS [12] as a measurement tool was the key principle in [10, 28]. 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.

## 7 Conclusions

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 is the only Internet-scale system that provides an infrastructure-free mechanism to estimate distance between remote hosts, *i.e.*, without any requrement for a measurement infrastructure nor for the manpower to administer the same. 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 demonstrated Fury Route's ability to construct chains to over 80% of destinations in the median case. We further showed that it matches the accuracy of infrastructure-dependent systems such as iPlane. We examined the potential for caching, showing a significant capability for caching route graphs, rapidly building chains with fewer than 50 queries. 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.

## References


1. Amazon Route 53, `https://aws.amazon.com/route53/`
2. Azure DNS, `https://azure.microsoft.com/en-us/services/dns/`
3. Dyn DNS, `http://dyn.com/dns/`
4. ECS dnspython, `https://github.com/mutax/dnspython-clientsubnetoption`
5. GoDaddy: DNS, `https://www.godaddy.com/domains/dns-hosting.aspx`
6. Google Cloud Platform: Cloud DNS, `https://cloud.google.com/dns/`
7. Neustar DNS Services, `https://www.neustar.biz/services/dns-services`
8. Verisign Managed DNS, `http://www.verisign.com/en_US/security-services/dns-management/index.xhtml`

9. Which CDNS support EDNS-client-subnet, https://www.cdnplanet.com/blog/which-cdns-support-edns-client-subnet/
10. Calder, M., Fan, X., Hu, Z., Katz-Bassett, E., Heidemann, J., Govindan, R.: Mapping the expansion of Google's serving infrastructure. In: Proc. of IMC '13 (2013)
11. Choffnes, D., Bustamante, F.: Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems. In: Proc. of SIGCOMM '08 (2008)
12. Contavalli, C., van der Gaast, W., tale, Kumari, W.: Client subnet in DNS queries(IETF draft) (December 2015), http://www.ietf.org/internet-drafts/draft-ietf-dnsop-edns-client-subnet-06.txt
13. Costa, M., Castro, M., Rowstron, A., Key, P.: PIC: practical Internet coordinates for distance estimation. In: Proc of ICDCS '04 (2004)
14. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: A decentralized network coordinate system. In: Proc. of SIGCOMM '04 (2004)
15. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A global Internet host distance estimation service. IEEE/ACM ToN 9(5) (Oct 2001)
16. Gummadi, K., Saroiu, S., Gribble, S.: King: Estimating latency between arbitrary Internet end hosts. In: Proc. of Internet Measurement Workshop (IMW) (2002)
17. Kuhrer, M., Hupperich, T., Bushart, J., Rossow, C., Holz, T.: Going wild: Large-scale classification of open DNS resolvers. In: Proc. of IMC '15 (2015)
18. Madhyastha, H.V., Anderson, T., Krishnamurthy, A., Spring, N., Venkataramani, A.: A structural approach to latency prediction. In: Proc. of IMC '06 (2006)
19. Madhyastha, H.V., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., Venkataramani, A.: iPlane: An information plane for distributed services. In: Proc. of OSDI '06 (2006)
20. Micka, S., Goel, U., Ye, H., Wittie, M.P., Mumey, B.: pcp: Internet latency estimation using CDN replicas. In: Proc. of ICCCN (2015)
21. Ng, T., Zhang, H.: Predicting Internet network distance with coordinates-based approaches. In: Proc. of IEEE Infocom '02 (2002)
22. Otto, J.S., Sánchez, M.A., Rula, J.P., Bustamante, F.E.: Content delivery and the natural evolution of DNS: Remote DNS trends, performance issues and alternative solutions. In: Proc. of IMC '12 (2012)
23. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP throughput: A simple model and its empirical validation. In: Proc. of SIGCOMM '98. Vancouver, British Columbia (Sep 1998)
24. Rabinovich, M., Triukose, S., Wen, Z., Wang, L.: DipZoom: The Internet measurements marketplace. In: Proc of IEEE Infocom '06 (2006)
25. RIPE Atlas, https://atlas.ripe.net/
26. Sánchez, M.A., Otto, J.S., Bischof, Z.S., Choffnes, D.R., Bustamante, F.E., Krishnamurthy, B., Willinger, W.: Dasu: Pushing experiments to the Internet's edge. In: Proc. of USENIX NSDI (2013)
27. Shavitt, Y., Tankel, T.: On the curvature of the Internet and its usage for overlay construction and distance estimation. In: Proc. of INFOCOM '04 (2004)
28. Streibelt, F., Böttger, J., Chatzis, N., Smaragdakis, G., Feldmann, A.: Exploring EDNS-client-subnet adopters in your free time. In: Proc. of IMC '13 (2013)
29. Su, A.J., Choffnes, D., Bustamante, F., Kuzmanovic, A.: Relative network positioning via CDN redirections. In: Proc. of ICDCS '08 (2008)
30. Su, A.J., Choffnes, D., Kuzmanovic, A., Bustamante, F.: Drafting behind Akamai (Travelocity-based detouring). In: Proc. of SIGCOMM '06. Pisa, Italy (Sep 2006)
31. Wong, B., Slivkins, A., Sirer, E.G.: Meridian: A lightweight network location service without virtual coordinates. In: Proc of SIGCOMM '05 (2005)