# Rediscovering Recurring Routing Results

Xiao Song
University of Southern California
United States
songxiao@usc.edu

John Heidemann
University of Southern California
United States
johnh@isi.edu

## ABSTRACT

Routing is central to networking performance, including: (1) latency in anycast services and websites served from multiple locations, (2) networking expenses and throughput in multi-homed enterprises, (3) the ability to keep traffic domestic when considering data sovereignty. However, understanding and managing *how* routing affects these services is challenging. Operators use Traffic Engineering (TE) with BGP to optimize network performance, but what they get is the result of *all BGP policies* throughout the Internet, not just their local choices. Our paper proposes *Fenrir, a new system to rediscover recurring routing results*. Fenrir can *discover* changes in network routing, even when it happens multiple hops away from the observer. Fenrir also provides new methods to *quantify* the degree of routing change, and to *identify* routing "modes" that may reappear. Second, we show that Fenrir can be applied to many different problems: we use five instances of three different types of systems to illustrate the generalization: anycast catchments showing in a root DNS service, route optimization for two multi-homed enterprises, and website selection for two of the top-10 web services. Each type requires different types of active measurements, data cleaning and weighting. We demonstrate Fenrir's methods of detecting and quantifying change are helpful because they all face similar operational questions: How much effect did traffic engineering have? Did a third-party change alter my routing? In either case, is the current routing new, or is it like a routing mode I saw before?

## 1 INTRODUCTION

Routing has been a core component of the Internet since its inception [15]. Since the early 1990s, routing meant finding a good available path and quickly selecting an alternate after a routing failure. As the Internet commercialized in the mid-1990s, route *optimization* grew in importance [44], often to minimize cost while considering performance.

**The Need to Understand Routing and Optimization:** The growth of hypergiants (such as Google and Facebook [31]) has made route optimization mandatory, since egress traffic can overwhelm individual links [28, 51, 63], and minimizing network latency is often a competitive advantage. With their global backbones (for example, [28]), and extensive global peering mixing transit providers with public IXPs [5] and private interconnects [39] at hundreds of points-of-presence (PoPs) [11], route optimization has huge effects on hypergiant operational expenses and performance for their customers. The effects of traffic shifts due to the growth and broad-peering hypergiants has been described as "the flattening of the Internet", a trend in Internet routing [14, 31].

Routing also drives IP Anycast operations, determining *Anycast catchments*, which traffic goes to which physically distributed anycast sites. Today, the broad use of IP Anycast by DNS [16, 25, 42] and CDNs [29, 53] to minimize latency [52] and manage DDoS [35, 49].

Routing's role in business competition prompts network operators to understand how routing affects their customers, and to quickly detect routing changes. Route optimization tools are commercially available, and hypergiants often deploy custom tools (for example, [51, 63]). Users of IP Anycast have developed special tools to evaluate destinations [18, 56, 62]. Companies (ThousandEyes, Cisco, Dyn) and and non-profits (RIPE) monitor routing as a service, often with thousands of distributed observers [48]. Finally, public routing data has been available for 20 years [45, 60]. Interpreting this data and improving routing remain important research topics.

Despite its importance, *today, there is no good way to summarize routing for a service, nor to differentiate between tiny changes and huge shifts*. It is challenging to understand routing, even for a single destination, because no one organization has a global view of routing—routing emerges from combining the Internet's diverse routing policies as determined by business and policy constraints.

Of course, service operators know about routing changes they initiate, and their transit providers may notify them of changes in their immediate upstream. But routing is a function of *everyone in the Internet*, policy or operational changes multiple hops away can affect service performance. One specific recent example is unexpected cable cuts in the Baltic Sea [41], resulting in latency seen in some European networks [3]. These changes were explained with a one-off research analysis, but in general operators have *no direct way* of detecting how much routing changes beyond their immediate neighbors affect their reachability.

**Contributions:**

The first contribution of our work is to present *Fenrir, a new approach to rediscover recurring routing results*. Fenrir

takes inputs from measurement systems of distributed systems running across the Internet that are affected by routing, cleans and distills information into *routing results* that summarize state of the service in terms of *catchments* and relate to how networks access the service. We quantify differences between vector pairs over the entire time series, and automatically cluster vectors to find mostly stable "modes" in routing results. We visualize those differences to help operators reach actionable decisions.

Our second contribution is to show that *Fenrir can apply to various network services* where routing plays a critical role. We evaluate customer assignment to sites in IP anycast systems with the B-Root DNS service. We examine routing out a multi-homed enterprise, the University of Southern California, examining both immediate upstreams and our upstream's transit providers. Finally, we examine Google's website, a top-10 website that is served by thousands of front-ends. We show that Fenrir can also help make sense of Google's DNS-based redirection, in addition to routing. Fenrir applies to these different services, each poses unique data collection and cleaning needs.

Although these services differ in detail, they are common in each, *Fenrir helps service operators answer important operational questions.* How quickly do catchments change? Do routing results re-occur at later times? Are two 80% similar, or 40%? How often do third-party routing changes (perhaps multiple hops upstream) change catchments? While we directly report about routing, network operators can combine routing with service-specific information such as traffic load or customer latency to quickly understand the implications of intentional and third-party routing changes.

The third contribution of this paper is to *study these real-world systems to validate Fenrir accuracy, and to see what Fenrir shows about each.* We validate Fenrir against ground truth from B-Root operators (§3), proving that Fenrir sees nearly all known routing changes, with recall of 1 and accuracy of 0.84. We also show data that suggests that third-party routing changes have visible effects on service-specific catchments.

We examine what Fenrir finds in our three services of interest (§4). Based on eight-month data of the multi-homed enterprise USC, we show a significant routing change indicating that at most 90% of catchments have changed. By evaluating the five-year data of B-Root, the routing is relatively stable; adding new anycast sites and removing old sites result in new modes. Also, by comparing the routing results at the end of 2019 and the end of 2024, we observe that around 30% of networks fall back to previous routing mode. From two months of data for Google, we confirm their aggressive policy for service evolution. From one and half months of data for Wikipedia, we observe that one site was drained and later up again, however, the new routing result is only 80% similar to the previous one.

**Data availability:** Our paper uses a combination of publicly available data and new datasets we collect, as shown in Table 2. In addition to existing public RIPE Atlas data, we will release our enterprise and top-website datasets to researchers at no cost, with the publication of our paper.

**Ethics:** Our paper studies network routing and its impact on public-facing services, so our data poses no privacy concerns to individuals. We do collect data to and from millions of networks, but we have no knowledge of which networks are used by specific individuals, and we will not join our data with user-to-location mapping data, nor allow others to do so, as required by a usage agreement.

## 2 METHODOLOGY

Fenir's process is summarized in Table 1. We actively collect data that shows how routing affects the observer. After cleaning, it generates vectors that represent routing results. We show how to compare vectors to quantify how similar routing is on different days, and to discover and visualize significant changes.

### 2.1 Problem Statement

Routing establishes paths from *global networks* to *services.*

To capture how routing affects a service, we define *catchments* as cut (in the sense of graph theory) between those networks and that service. Our use is a generalization of how catchment is used in IP Anycast, where catchments are the list of server sites and reflect which client networks visit each site (and, in turn, from hydrography, where catchments are the drainage basins of rivers).

Our generalization considers not only catchments at the service, but also in the *middle* of the network. For example, for a multi-homed enterprise, catchments can represent the path upstream provider used to reach each destination network at each hop. Understanding catchment evolution is important and challenging because they depend not only on routing policies at the source and destination, but also by the *routing policies of parties in the network.*

We aim to summarize the catchment state in *routing results.* Mathematically, a vector contains all information about a service's catchments at a specific time. It defines a particular routing result, and allow operators to determine what changes and if a routing result re-occurs. We *compare* vectors to quantify degrees of similarity or difference; operators would like to know if routing is "80% like last month". Operators often track user-relevant performance metrics, so a 20% similar vector may require re-investigation of performance gains and losses, but if a route change shows a vector that

| Step | Goal | How |
|---|---|---|
| Identifying subjects (§2.3) | Define target networks and destinations | Public datasets |
| Data collection (§2.3) | Observe associations between networks and service cachements | Active probing |
| Data cleaning (§2.4) | Remove bogus data; fill missing holes | Service-specific |
| Comparison function (§2.6.1) | Direct pair-wise comparison among pairs and timeseries | Gower's distance |
| Clustering (§2.6.2) | Find routing "modes" | Hierarchical Clustering |
| Quantification (§2.7) | Show changes and similarities | Heatmap and transition matrix |
| Performance (§2.8) | Evaluate performance of routing modes | Latency |

**Table 1: Our steps to rediscover recurring routing results.**

| case study | service | catchment | network | dataset: name | start | duration |
|---|---|---|---|---|---|---|
| anycast | DNS or anycasted services | anycast sites | 5M IPv4 /24 blocks [18] | B-Root/Verfploeter | 2019 | 5 years |
| | DNS or anycasted services | anycast sites | 13k RIPE Atlas VP [48] | B-Root/Atlas | 2019 | 5 years |
| multi-homed enterprise | an enterprise | upstream providers | 1.6 M IPv4 /24 blocks | USC/traceroute | 2024-08 | 8 months |
| top websites | a hypergiant website | website instances | global networks | Google/EDNS-CS | 2024-02 | 2 months |
| | a non-profit website | website instances | global networks | Wiki/EDNS-CS | 2025-03 | 1.5 months |

**Table 2: Paper terms (network, catchment, service) and datasets used for three different systems.**

is 95% like last quarter, the operator will have confidence in what performance to expect even before a new investigation.

Routing affects Internet services and traffic in general, and we expect Fenrir can be applied to several problem domains. Table 2 lists three classes of systems and six combinations of measurement methods. We study 1) traffic routing from a multi-homed enterprise such as a university; 2) anycast catchments, as are widely used by large DNS providers [4, 8] and Content Delivery Networks [6, 11]; 3) websites for major Internet properties, like Google or Facebook [11, 20, 26, 51] and for non-profit organization, like Wikimedia that hosts Wikipedia. All of these services share a common need to understand and influence routing. Multi-homed enterprises seek to maximize performance for users while minimize traffic across peers which may be no-cost, fixed cost, or metered. DNS providers and CDNs strive to lower latency [9, 12, 21, 29, 32, 33, 50, 52, 61, 64] and manage traffic during large DDoS events [35, 37, 49]. Major Internet services seek to balance ingress and egress traffic, not overloading links and maximizing user QoE (for example, [51]). Organizations such as RIPE evaluate country-level Internet access [47] with metrics such as AS-hegemony [22]

**A Case Study:** As an example illustrating our goals, Figure 1 shows ten days of anycast catchments for G-Root. Each of the five different colors represents and how many observers are sent to each anycast site (the anycast catchment) for five sites (one is not observed) and error and other responses.

This visualization helps us detect operationally important changes. Around midnight on 2020-03-03, STR (in Stuttgart, Germany) almost completely drains, with its users shifting to NAP (in Naples, Italy). This situation reverts 4.5 h later.

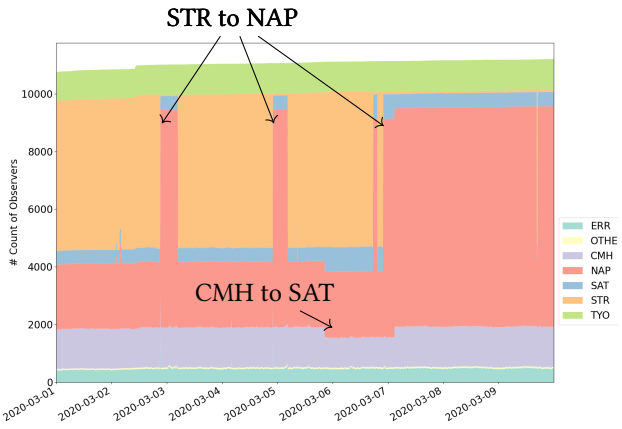



**Figure 1: Catchment sizes in G-Root, as measured by counts of RIPE Atlas VPs (measurement id 10314). 2020-03-01 to 2020-03-09.**

This same mode happens again on 2020-03-05, and a third time, on 2020-03-07, it shifts again and remains through the end of observation. With one site completely draining, we suspect these changes reflect maintenance activity by the service operator.

We also see a second, smaller shift of users from CMH to SAT for two days, starting on 2020-03-06. This event could represent intentional route changes by the operator, or it could be changes resulting from some third-party network's routing policy.

Our goal is to assist operators in discovering these changes and understanding how affect service: how many users shifted after maintenance? What was the performance of the different maps? Were the shifts of users the same of different on different days? If the secondary event was caused by a third party, were operators aware of it? Did it affect performance for clients that kept the same catchment?

## 2.2 Defining the Vector

In Table 2, our goal is to relate how networks use services, with cachements representing a cut through the network graph. We formalize a vector at time $t$ as $D(t)$, with one element for each of the $N$ networks. Each element takes one of $|S|$ values, one for each possible service site in $S$, as $S$ is the set of service sites. As an example, part of the vector for Figure 1 on 2020-03-01 might be:

$$D = [\text{CMH}, \text{NAP}, \text{STR}, \text{STR}, \text{OTHER}, \text{SAT}, \text{ERR}, \dots]$$

It means that the first network goes to CMH, the second network goes to NAP, the third and forth network go to STR, etc.

A one-hot representation is helpful mathematically, where we define a normalized vector, as the matrix $D_*(t)$, size $N \times |S|$, where each cell $D_*(t, n, s)$ is 1 if $D(t, n) = s$, otherwise 0.

We can summarize how many networks go to each site based on aggregating by site into an $|S|$-long vector $A(t)$.

$$A(t, s) = \sum_{\forall n \in N} D_*(t, n, s)$$

Continuing our example from 2020-03-01 in Figure 1, we get an aggregated vector $A$ for sites CMH, NAP, STR, NRT, SAT, HNL, err, and other.

$$A(\textit{2020-03-01}) = [1350, 2200, 5200, 1000, 480, 10, 560, 50]$$

Two days later on 2020-03-05, when STR mostly drains:

$$A(\textit{2020-03-05}) = [1350, 7300, 1, 1000, 480, 10, 440, 50]$$

## 2.3 Data Collection

We use active measurements to determine the current routing result. The exact protocol used varies by the service being studied, with the requirement that it be able to indicate the catchment each network is in. Next, we describe measurement protocols for each service next, and then how we scale raw observations to normalize coverage. Table 2 lists our datasets and measurements.

*2.3.1 Anycast.* Anycast catchments are defined by the anycast site for each client to reach, as determined by BGP routing. We use two methods to map anycast catchments: RIPE Atlas observes which DNS root server they reach from each of their about 10k vantage points (VP) [46]. RIPE Atlas has archived such measurements for more than a decade. Atlas uses standard DNS mechanisms (CHAOS queries to hostname.bind or NSID [62]) to identify a per-server identifier. We then map these organization-specific identifiers to determine specific anycast sites, following prior work [19]. Figure 1 shows an example of catchments at G-Root using this method.

Alternatively, we measure anycast catchments from an anycast system using Verfploeter [18]. Verfploeter requires the involvement of the anycast system, and the B-Root operators have shared about three years of data with us. Verfploeter provides broader coverage than Atlas, associating 5M networks with their B-Root destinations. Verfploeter gets this coverage, pinging targets in millions of networks and watching which catchment the reply goes to to determine the target's current catchment. Figure 3 shows Verfploeter data for B-Root.

*2.3.2 Network connectivity.* Cachements in network connectivity are defined by what upstreams or transit providers one uses to reach destination networks. We map enterprise upstream catchments by taking traceroutes out of the enterprise to all routable network prefixes. Then, we look at a reasonable distance outside campus, but not infinitely far away. RIPE maps country-level transits with control-plane observations: they observe routing from RIPE RIS route collectors, and look through the AS paths in the BGP routing table to all IP blocks in the country under study.

Unlike anycast catchments, which are defined by the anycast site each network is assigned to, network connectivity can consider entire paths. While enterprises are usually concerned about their immediate upstreams, and RIPE country-level analysis studies Tier-1 transit providers out of the country, in both cases, one can adjust the "focus" of the study to consider more or fewer hops.

We use scamper [34] to take traceroute from one server at USC, measuring the first 10 hops to selected routable addresses based on a hitlist [19] of every /24 blocks. We cover 1.6M /24 networks by comparing hitlist and routable blocks obtained from the BGP table of RouteViews. We terminate Traceroute when it exceeds the 10th hop. It takes around 8 hours to complete a full list scan. The probing rate is 550 packets per second. We assume the underlying routing does not change rapidly, keeping the probing rate low to reduce the stress on the first hop while being able to capture the changes.

Figure 2b is an example of routing out of one multi-homed enterprise, USC.

*2.3.3 Top Websites.* Like anycast, catchments in top websites are defined as the site that serves the request for each network that originates a query. We measure top websites by making queries to URLs for the front page of a website. Most websites use DNS-based load-balancers that select specific

servers near the target, so website evaluation involves both DNS and HTTP queries from different locations.

We want to understand why website front-ends are chosen for all potential global observers. Since no one has observers in all networks, we use the EDNS/Client-Subnet approach from Calder et al [11]. To evaluate the catchment from a given observer $N$, we make a DNS lookup for the hostname in the website URL given the prefix $N$ as the client-subnet option. For websites that use DNS-based server selection, if the DNS recursive resolver passes Client-Subnet through, we can map global catchments using a single physical observer.

We studied two targets using this method: Google, with thousands of front-ends aggressive churn, and Wikipedia, with seven global sites and more modest changes. Figure 5 shows Google rapidly changing routing, and Figure 6b shows a significant routing change at Wikipedia on 2025-03-19.

## 2.4 Data Cleaning

Although active measurements provide input to our algorithms, such raw observations often have errors or gaps, therefore we clean raw observations in three ways.

**Remove Incorrect Data:** Some measurements contain incorrect data. The details vary by service, but when we identify clearly incorrect data, we discard it.

**Remove Micro-catchments:** Some services have many sites, but not all are equally important. We therefore identify *micro-catchments*, which are defined to be sites that are responsible for few newtorks, as determined by what we observe in our observation system. Removing micro-catchments reduce the complexity of identifying routing results. It helps us focus on large catchments that would significantly affect routing.

Examples of micro-catchments occur in anycast and enterprise routing. In anycast, local-only sites serve only a single AS and its customers, not the general Internet. If no VPs are observed from within that AS, catchments for local sites will be of zero size. Second, in enterprise routing, we are most concerned about how the enterprise reaches the world. However, the enterprise may have routes to a few local network prefixes, such as networks within it. We can filter out such networks if desired.

**Interpolate Missing Data:** One-shot active measurements often miss data due to random packet loss. Even with retries, temporary network anomalies can result in data gaps. Missing values are problematic as they may skew the data towards a lower availability [27]. We fill gaps by interpolating data from recent successful observations. The details of what kind of gaps occur and how we fill them vary by service.

Verfploeter sends queries from a central site; if the query target is temporarily unavailable, that block will have no known catchment. We fill missing anycast observations by replicating the most recent successful observation. The same cleaning strategy applies to top website measurements with EDNS/CS as well.

Measurements of enterprise routing with traceroute provide redundancy as each hop is queried multiple times, but intermediate traceroute hops with private addresses or that filter pings will not provide results. Since traceroutes are often successful at some hops even if unsuccessful at others, we use this spatial redundancy and propagate the nearest viable hop to fill a traceroute gap.

We adopt nearest neighbor imputation to fill up the hole of missing values. Specifically, we address the losses of consecutive probes by identifying two positive responses separated by a series of non-responses. We then fill this gap with positive responses from our nearest neighbors. For example, a list of consecutive probes is denoted by $[1, \ldots, n]$. Assume a series of consecutive probes $[k, \ldots, k+i]$ is missing. To fill up missing holes, all probes in $[k, \ldots, k+i/2]$ are replaced by $[k-1]$ and all probes in $[k+i/2, \ldots, k+i]$ are replaced by $[k+i+1]$. We put a limit (up to 3 observations away) on $k$ to how far we can interpolate the missing value to avoid changing vectors.

## 2.5 Weighting

Once cleaned, our measurements provide a preliminary list of of catchments that associate networks with services. While true, this list may not represent metrics that are most operationally important—each observation indicates what that VP sees. What operators care about is what that VP *represents*—how many users are there, or how much traffic is sent from that location. We therefore adjust raw observations with a service-specific *weighting factor* to represent metrics of operational interest, like IP addresses, traffic, or users. We therefore assign a weight vector $D_w(t)$ that parallels the vector, showing how "important" each component is. A default weight vector might be all 1, showing each observation is equivalent, but alternatives might weigh observers differently, considering the number of addresses, users, or traffic.

For anycast, we can weight results using networks and traffic estimates. Observers with RIPE Atlas and Verfploeter may not be uniformly distributed across the IPv4 address space, so, as our first step, we can weight each observation by how many networks it represents. Thus, if we have only one Atlas VP or Verfploeter response from a /16 prefix, we can count that as 256 /24 blocks rather than just one. We make the assumption that addresses in the same /24 blocks are located in the same or nearby physical location.

For services that collect and summarize historical traffic or users we can go one step further and weight blocks by the amount of historical traffic they have sent, or users they have.

Verfploeter compared addresses and traffic for DNS [18]. Since network occupancy and user traffic demands vary considerably, so can these estimates.

Enterprise and country-level routing also benefits from weighting by IP addresses and traffic estimates.

Top websites should be weighted by the number of users in each network. Although we do not have access to user distributes and their traffic demands, certainly websites have this information.

## 2.6 Analysis: Comparion and Clustering

Given a vector $D$ composed of clean data and weights ($D_w$), we next describe how to compare and cluster vectors to discover trends.

*2.6.1 Pairwise Comparision.* We first define a pairwise comparision function to judge how similar any two vectors are.

For pairwise comparion, we adopt Gower's Distance [24] between all $N$ elements. When comparing elements at times $t$ and $t'$ match if they are part of the same catchment:

$$M(t,t',n) = \begin{cases} 1, & \text{if } D(t,n) = D(t',n) \wedge D(t,n) \neq \text{unknown}, \\ 0, & \text{otherwise.} \end{cases}$$

And the normalized, weighted similarity $\Phi(t,t')$ of Gower's Distance is:

$$\Phi(t,t') = \frac{\sum_{n \in N} M(t,t',n) D_w(n)}{\sum_{n \in N} D_w(n)}$$

$\Phi$ has a physical meaning: it is the fraction (from 0 to 1.0) of networks that are the same between two vectors. We plot distances between all-pairs combinations of vectors. using a method we describe below §2.7, providing heatmaps such as Figure 5.

The test for "unknown" in $M(t,t',n)$ treats all cases where we do not know network $n$'s catchment as changing. This assumption is pessimistic and reduces $\Phi$ for services where observations are imperfect. Measurements are frequently imperfect for Verfploeter measurements of catchments since Verfploeter requires a ping response to confirm catchment, but predicting a responsive IP address in a target network employing dynamic address assignment is probabilistic. Since Verfploeter reports unknown for about half of it is 5M target networks, a stable catchments will show $\Phi$ values between 0.5 and 0.6, rather than near 1.0. (As ongoing work, we plan to remove unknown networks from consideration when computing $\Phi$, so it will define similarity of *known* networks.)

*2.6.2 Clustering.* The comparison allows us to visualize the similarity of different vectors, but to automatically discover similar groups we use Hierarchical Agglomerative Clustering (HAC), a method of unsupervised machine learning [55]. HAC merges vectors into clusters $\mathbf{T}_c$, the set of times where vectors $D(t) \forall t \in \mathbf{T}_c$ are similar, using a bottom-up strategy.

HAC will continue clustering until all vectors are in one cluster, so we define a distance threshold to indicate when clusters are different enough to be considered separate.

We determine the Distance Threshold adaptively. One challenge is that entities are operated individually and have a different number of target networks and destinations. To find the best distance threshold for different entities, we loop over a range of distance threshold $[0, 1]$ with step 0.01 and construct a new HAC model with the distance threshold. We choose the first HAC model with less than 15 clusters with at least 2 valid observations. In practice, we note that the number of clusters converges quickly when the distance threshold increases. One can tune the parameters to fit its needs better.

## 2.7 Quantifying Differences Between Vectors

While a vector summarizes the current routing result, and our normalized Grower's distance compares any two vectors, operators need to compare *many* vectors to highlight periods of routing stability and the degree of change.

We summarize routing over time by comparing all pairwise vectors as a gray-scale heatmap, such as Figure 5. Heatmaps make it easy to identify blocks of similar routing results as high-similarity (dark-shaded) triangles, and changes as discontinuities in shading.

We sometimes augment heatmaps with a *transition matrix* to compare two vectors and summarize how they are different. A transition matrix $T$ is an $|S| \times |S|$ matrix, where each $T(t,t',s,s')$ element evaluates how many networks were in state $s$ in vector $D(t)$ and are in state $s'$ in vector $D(t')$.

$$T(t,t',s,s') = \sum_{n \in N} 1 \text{ if } (D(t,n) = s \wedge D(t',n) = s')$$

Let $\Psi_{t,t'}$ to be a transition matrix of $N$ target networks and $V$ destinations between time $t$ and time $t'$. $\Psi_{t,t'}[c_v, c_u]$ denotes the number of target networks leaving from destinations $c_v$ to destination $c_u$ from time $t$ to time $t'$, $t, t' \in T$, $c_v, c_u \in V$.

For a completely quiescent network, $T(t,t')$ will be a diagonal matrix equal both to $A(t)$ and $A(t')$. However, when networks change the catchment between the two vectors, they contribute to cells off the diagonal. In Table 2a, we see that 3097 networks move from STR to NAP (in yellow), and 1542 move from STR to the error state (they get no replies from any site). In the next four minutes, Table 2b, we see

**(a) Large shift from STR to NAP, from 21:56 to 22:00.**

| initial state \ subsequent state | CMH | NAP | STR | NRT | SAT | HNL | err | oth |
|---|---|---|---|---|---|---|---|---|
| CMH | 1352 | 0 | 0 | 1 | 0 | 0 | 16 | 0 |
| NAP | 0 | 1939 | 0 | 1 | 0 | 0 | 272 | 0 |
| STR | 0 | 3097 | 625 | 4 | 0 | 0 | 1542 | 0 |
| NRT | 1 | 2 | 0 | 985 | 0 | 0 | 30 | 0 |
| SAT | 1 | 0 | 0 | 1 | 472 | 0 | 2 | 0 |
| HNL | 0 | 0 | 0 | 0 | 0 | 12 | 1 | 0 |
| err | 17 | 45 | 15 | 14 | 7 | 0 | 309 | 0 |
| oth | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 46 |

**(b) Drain of STR completes, from 22:00 to 22:04.**

| initial state \ subsequent state | CMH | NAP | STR | NRT | SAT | HNL | err | oth |
|---|---|---|---|---|---|---|---|---|
| CMH | 1344 | 0 | 0 | 1 | 1 | 0 | 20 | 0 |
| NAP | 0 | 4987 | 0 | 0 | 0 | 0 | 87 | 0 |
| STR | 0 | 636 | 2 | 0 | 0 | 0 | 11 | 0 |
| NRT | 1 | 0 | 0 | 993 | 0 | 0 | 8 | 0 |
| SAT | 0 | 0 | 0 | 0 | 473 | 0 | 7 | 0 |
| HNL | 2 | 24 | 0 | 0 | 0 | 12 | 0 | 0 |
| err | 17 | 1801 | 0 | 35 | 2 | 1 | 317 | 1 |
| oth | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 45 |

**Table 3: Transition matrices for G-Root on 2024-03-04. Sites are airports, plus error and other states.**

the transition completed and STR nearly completely drains, with most sites moving from error to NAP.

## 2.8 From Similarity to Performance Differences of Vectors

While heatmap identifies regions of similarity and transition matrix specifies how any two vectors are different, operators care about *user relevant metrics*, not just how many routes changed. *Latency* is an important metric, so we would like to estimate *latency differences between pairs of vectors*

We therefore factor latency into our vectors to allow operators to estimate the effects a routing change has on latency and if they should investigate further and take some response. We compute mean overall latency by measuring latency for each network and weighting that value by predefined matrix §2.5. As with data collection, we use different data sources to estimate latency for each service.

*2.8.1 Anycast.* Many measurement systems do probing to collect RTT toward the anycast sites. For instance, RIPE Atlas takes data from its 13k VPs to all Root DNS with built-in measurements (§2.3.1). Each response contains RTT, representing the latency from the end host to the DNS anycast site.

*2.8.2 Network connectivity.* Measuring latency experienced by multi-homed enterprises is tricky, as collecting end-to-end RTT requires scanning the whole Internet from the enterprise network. In this case, we use Trinocular [43], an existing Internet outage detection system operating since 2013. One of Trinocular sites sits inside USC; it scans approximately 5.2 million /24 IPv4 address blocks using ICMP echo request messages. Each site probes between 1 and 16 targets per block every 11 minutes, selecting targets from a fixed pseudorandom list updated quarterly.

Both data sources provide frequently updated RTT from VPs to the measured network, with at least one site controlled by us, satisfying our needs and without running extra measurements.

# 3 VALIDATION

To validate our work, we compare changes detected by Fenrir with the ground truth: verified operational data provided by the service operator.

**Ground truth:** We assess Fenrir's ability to rediscover recurring routing results by comparing its detected changes in B-Root catchments against operator-provided ground truth over a four-month period beginning on 2023-03-01. Fenrir identifies events in the B-Root/Atlas dataset by examining transitions in vector matrices every four minutes. Many operator-reported events are short-lived, often lasting only tens of minutes. They would not appear in datasets collected at lower temporal resolution, such as the daily B-Root/Verfploeter data.

The ground truth is derived from B-Root operator maintenance logs. Across this period, we observe 98 recorded maintenance entries. Because some maintenance activities are the combination of several events, where some are externally visible and others not, we group events occurring within ten minutes and performed by the same operator into 56 event groups. We classify these as either invisible (internal changes with no observable external effect) or external (changes such as site drains or traffic-engineering adjustments that affect catchments). As an example, an invisible event would be taking one of several replicated internal servers offline temporarily, where service automatically transfers to other active servers. A site drain indicates an intentional, temporary removal of the site from anycast so that the router or firewall can be changed without query loss. raffic engineering denotes routing adjustments that preserve reachability but shift catchments. This operator-derived ground truth therefore captures only expected B-Root routing changes.

| ground truth | Fenrir observations detected in F. | not detected |
|---|---|---|
| all logged events | 56 (98 before grouping) | |
| external | 19 (TP) | 0 (FN) |
| site drain | 17 | 0 |
| traffic engineering | 2 | 0 |
| internal only | 8 (FP?*) | 29 (TN) |
| **external changes?** | 10 (*) | — |

**Table 4: Evaluation of ground truth changes against Fenrir-visible changes for B-Root/Atlas.**

**Comparison:** Table 4 compares ground truth (rows) against Fenrir observations (columns). Our first conclusion is that *Fenir has good accuracy 0.86* ($(TP + TN)/all$)) detecting most events, with 19 true positives and 29 true negatives.

*Fenir has perfect recall of 1.0* ($TP/(TP + FN)$), missing no events. Our first examination of the data showed one false negative, but confirmation of the operators revealed that that event did have maintenance, but without draining the site, making it an event that should be invisible to external detection.

Consideration of false positive events requires more care because *not all actual routing changes events are in ground truth.* As presented, Fenir precision is moderate at 0.70 ($TP/(TP+FP)$), because a direct comparison of ground truth against Fenrir's predictions shows eight false positives. However, we need to consider that Fenrir detects not only ground truth events, but *also* events that caused by routing changes by third parties, such as transit providers. Such changes will not appear in ground truth data, which considers only operator-initiated changes.

In fact, *detection of third party changes is Fenrir's design goal* so this seemingly imperfect precision relative to known maintenance events likely instead represents *10 third-party routing changes unknown to the anycast operator and therefore of their interest—new visibility provided by Fenrir.*

Some evidence to support this hypothesis is that we also see 10 events in Fenrir (marked with (*)) that have no reflection in operator logs. As ongoing work, we are currently looking at historic routing data to find explicit evidence for such third-party events.

# 4 RESULTS

We next examine how Fenrir can discover events in each of our target systems listed in Table 2: multi-homed enterprises, anycast, and top websites.

To measure multi-homed enterprises, we study the routing behaviors of the upstream ISP of the University of Southern California (USC). We describe our approach to collect data at §2.3.2. We have also observed a second enterprise for 10 months, but thus far, we have not seen significant routing changes.

As an example system using anycast, we measure B-Root, since we can get ground truth about routing changes from its operators. We measure it with one data source, Verfploeter [18], from the B-Root operators. Unlike data collected by RIPE Atlas as used in §3 Validation, which is run independently, Verfploeter must be run by the anycast service, but it provides data for the catchments selected by about 6.1M global networks. We explain the details at §2.3.1.

We measure two top-10 websites: First, Google home page, served from thousands of sites [11], and also Wikipeida's home page, served from 7 locations [1]. In each case we use EDNS/client-subnet to simulate requests originating from 5M routable prefixes, as described in §2.3.3.

## 4.1 Changes Upstream from an Enterprise

Many large enterprises are *multi-homed*, connecting to multiple transit providers and other peers to send traffic to thousands of users to the global Internet. Large enterprises often peer directly with important services, like cloud providers or CDNs, or nearby peers and regional networks. University, for example, benefit from academic-networks (such as Internet2, GEANT, and ESNet) that are often both high-speed and low-cost. Traffic engineering and optimization can have a large impact on cost and performance. Such enterprises can optimize routing for cost and latency.

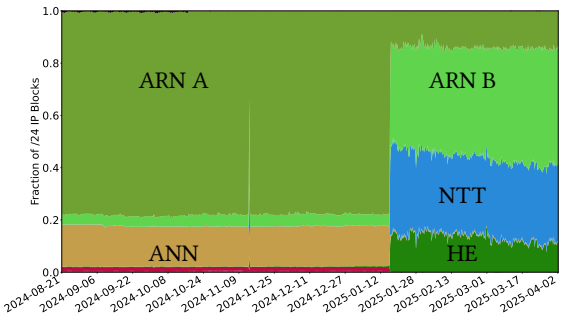
While an enterprise will directly monitor traffic at their borders, knows about their routing changes, and is likely informed about routing changes in their transit providers, *they are often unaware of changes that happen elsewhere in the Internet.* Yet such changes can be significant—unexpected cable cuts in the Baltic [41] resulted in latency changes in Europe [3], just one recent example of a change outside ones immediate peers with performance impact.

Fenrir has the potential to monitor and quantify routing changes that affect the path to any network. We next show how Fenrir analysis can digest traceroutes to millions of networks (as described in §2.3.2) to quantify changes to an enterprise's routing cone.

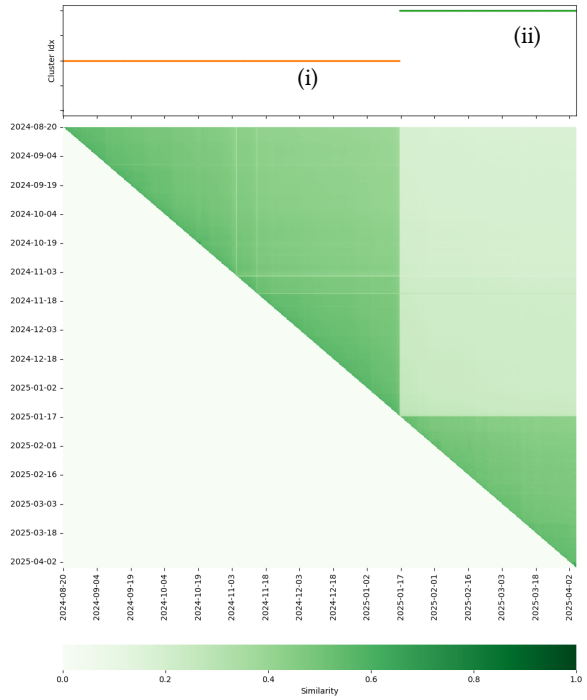
*4.1.1 Changes in Routing Modes:* Here, we show how many modes are detected, how different they are, and why.

We first consider where traffic is, three hops away from the enterprise. We use data covering eight months, starting 2024-08, for the University of Southern California.

Figure 2b shows Fenir heatmap with two strong routing modes: mode (i) with $\Phi$ in [0.31, 0.65] and mode (ii) with $\Phi$ in [0.42, 0.57]). seperated by 2025-01-16. Comparing $\Phi$ before and after the 2025-01-16 transition gives $\Phi(M_i, M_{ii}) = [0.11, 0.48]$, a huge routing change.




**(a) Fraction of each catchment. ARN A: Academic Regional Network A. ARN B: Academic Regional Network B. ANN: Academic National Network. HE: Hurricane Electric.**




**(b) Heatmap of pairwise comparisions ($\Phi(t, t')$).**

**Figure 2: Enterprise catchments from 2024-08 to 2025-04 at hop 3. The routing change happens on 2025-01-16. Dataset: USC/traceroute.**

While Fenrir's heatmap detects this change, we turn to the stack plot in Figure 2a, Here with each colored region counting the number of destination networks ($A(t)$ from §2.2, couting global /24 prefixes) that are handled by a given transit network.

We see that in 2024, almost all networks were served by CENIC (a regional academic network) and Internet2. There was a major routing change on 2025-01-06, 80% of networks are now served by LosNettos(a regional network), NTT, and Hurricane Electric, while Internet2 vanishes in hop 3.

We can also visualize the enterprise routing cone as a Sankey diagram, to see what networks traffic uses 2, 3, and 4 hops away. Due to page constraints, we show before- and after-change Sankey diagrams in an appendix as Figure 7 and Figure 8.

We believe this dramatic change was a network reconfiguration done by USC operators. However, we were able to discover it via Fenrir heatmaps, and use Fenrir's data to visualize it with stack plots and Sankey diagrams. For routing changes that occur multiple hops away, like the Baltic cable cuts, Fenrir's heatmap would be the only way an enterprise's network operators would be aware of the size of a change.

## 4.2 Anycast Changes at B-Root

DNS services often use IP Anycast to provide service from many locations. Anycast both reduces latency to clients by associating each with a nearby server, and helps protect during DDoS events [35]. Latency directly affects clients and is often used as a metric to compare service providers, so DNS operators monitor latency and routing [9, 52], since inefficient routing can result in high latency due to anycast polarization [36].

We next examine five years of B-Root DNS anycast data to demonstrate how our vectors help identify trends and changes, collected with Verfploeter (§2.3.1).

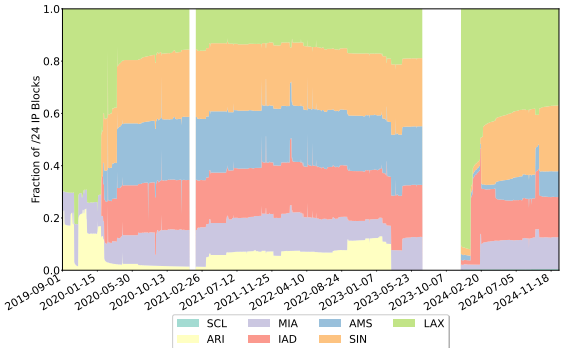
*4.2.1 Changes in Routing Modes.* Figure 3b shows a heatmap of vectors similarity over five years. In this graph, both axes show dates of observations, and each coordinate at $(t, t')$ compares the similarity of vectors between those dates. We automatically discover six common routing modes (mode (i) to mode (vi)) and these are visible as several darker triangles. (The blank region from 2023-07-05 to 2023-12-01 is a collection outage.)

To understand *why* these clusters appear similar, Figure 3a shows a stack-plot of the number of networks in each catchment.

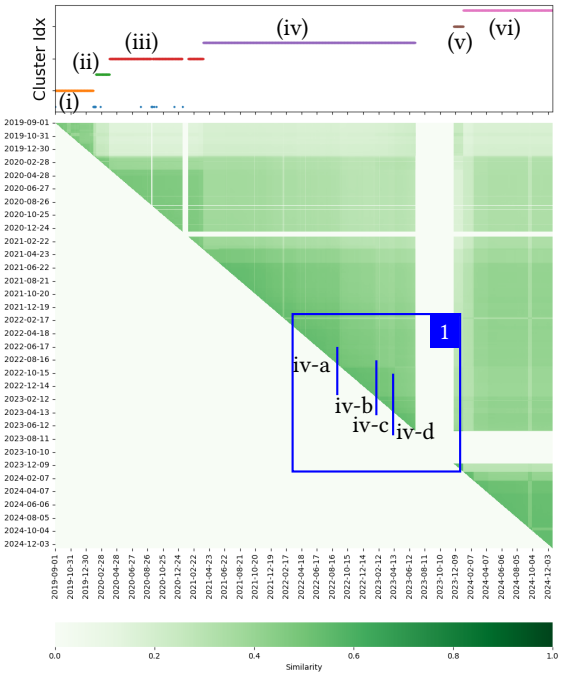
First, overall routing is relatively stable, sometimes for years at a time, with minor varations inside each cluster.

Routing is stable for the first five months, from 2019-09 to 2020-02 (mode (i), with the similarity $\Phi$ in [0.24, 0.49]).

There is a new mode (ii) from 2020-02 to 2020-04, with $\Phi$ in [0.26, 0.49], and $\Phi(M_i, M_{ii}) = [0.11, 0.48]$. This change corresponds to adding three new anycast sites SIN, IAD, and AMS to B-Root. We then see a different mode (iii), from 2020-04 to 2021-03, with $\Phi$ in [0.2, 0.54], and $\Phi(M_{ii}, M_{iii}) = [0.18, 0.48]$. This is becuase around 70% clients used to go




**(a) Fraction of each catchment.**




**(b) Heatmap of pairwise comparisions ($\Phi(t, t')$).**

**Figure 3: B-Root catchments from 2019-09 to 2024-12. Dataset: B-Root/Verfploeter. Zoom in the rectangle 1 in Figure 4 to show latency changes.**

LAX were routed to AMS, IAD and SIN. The longest-lasting mode (iv) covers 2021-03 to 2023-07, with $\Phi$ in [0.24, 0.49]. On 2023-07, routing changes from mode (iv) to mode (v), $\Phi(M_{iv}, M_v) = [0.18, 0.54]$. This corresponds with the move of ARI to a new location in the same country. Interestingly, mode (v) is somewhat like the original routing mode (i) in 2019-09, with $\Phi(M_i, M_v) = 0.31$, more so than its immediate neighbors ($\Phi(M_{iv}, M_v) = 0.22$, and $\Phi(M_v, M_{vi}) = 0.17$).

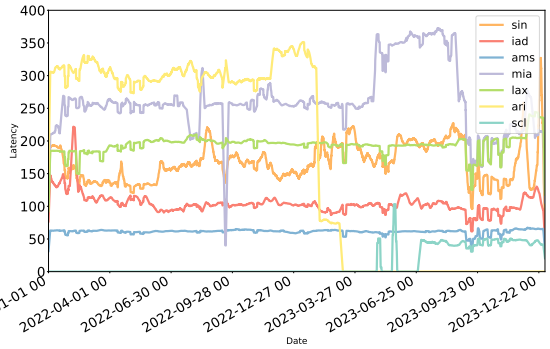



**Figure 4: The 90th percentile of latency of B-Root, from 2022-01-01 to 2023-12-31. Dataset: RIPE-Atlas**

This similarity is because LAX serves most clients in both (i) and (v). This ability to quantify similarity and discover connections such as between (i) and (v) is a Fenrir design goal.

The opportunity that routing modes provide to operators is to see trends over time, and to identify similarities in routing and differences. Similarities can give operators confidence in what latency their users will experience, and changes in similarity that are not accompanied by known changes by the operators suggest that some changes in third-party routing have affected their traffic. This global picture helps guide daily performance monitoring, providing routine that shows the "root causes" behind measurements of customer latency.

*4.2.2 Changes in Latency.* While operaters influence routing and catchments (§4.2.1), *latency to customers* is what really matters. We next show how similarities and changes in routing modes correspond to potential changes in latency.

As a case study, we examine B-Root from 2022-01 to 2023-12, the blue-boxed region [1] in Figure 3, which shows catchment changes for all networks as measured with Fenrir/B-Root/Verfploeter. Mode detection shows two modes over this period, (iv), followed by an unfortunate gap observation from August to October, then a new mode (v).

Close observation of the figure also shows several routing changes that are smaller than our threshold to declare a new mode. We mark these as (iv.a) to (iv.d) on Figure 3. We identify (iv.c) ending on 2022-09-16, (iv.b) ending on 2023-02-12, (iv.c) ending on 2023-04-13, (iv.d) ending on 2023-07-05. These events show up as small but significant changes in $\Phi$: each mode has an internal $\Phi$ around [0.475, 0.54], but with $\Phi(M_{iv.a}, M_{iv.b}) = [0.33, 0.54]$, $\Phi(M_{iv.b}, M_{iv.c}) = [0.39, 0.54]$, and $\Phi(M_{iv.c}, M_{iv.d}) = [0.4, 0.54]$.

To understand what these changes represent, we looked at public announcements about B-Root service changes [7], that service in South America was changing, with site ARI shutting down on 2023-03-06 and SCL starting in 2023-05.

We expect operators to watch Finrir, notice changes like those we note above, and for changes that are large enough, check on latency measurements. We therefore gather latency data for B-Root for each catchment net from RIPE Atlas to stand in for an operator's latency observations.

Figure 4 shows p90 latency for each of B-Root's catchments. We can see the service changes result in changes to catchments and latency: first the ARI catchment vanishes on 2023-03-06. ARI provided latency over 200 ms due to a few North American and European networks being routed to it, but its latency drops to zero when it is shut down. This After two months, SCL appears, but only briefly on 2023-05-01 and 2023-05-24. Our understanding from the discussion with operators is that these were temporary routing experiments as they were optimizing routing. Finally, on 2023-06-29, SCL resumed operation, providing very low latency.

This example shows how changes can be detected in Finrir via changes in Φ shown in Figure 3, and the operator uses them to trigger latency investigation (as we show in Figure 4). Although, in this case, the operator was aware of changes, if the changes had happened networks further upstream, Fenrir's ability to detect changes would be critical.

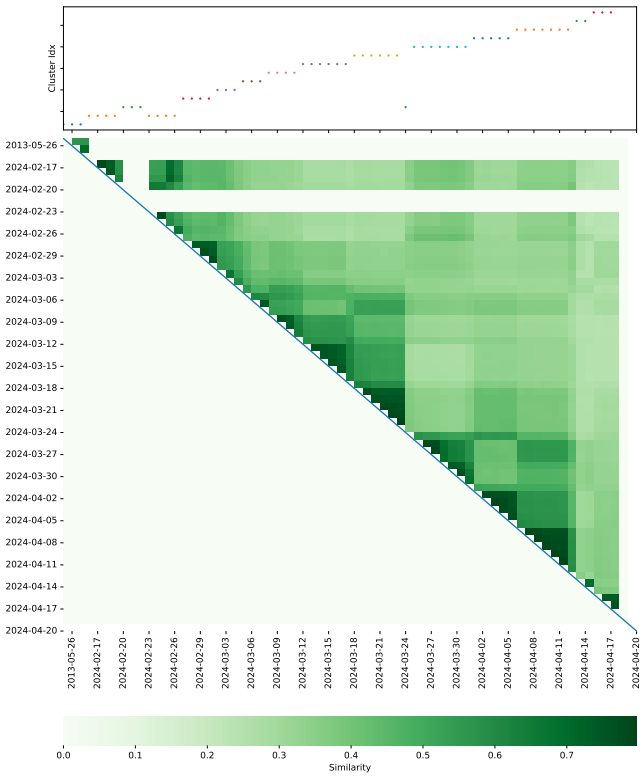



**Figure 5: Heatmap of routing changes of Google, from 2013-05-26 to 2013-05-28 and from 2024-02-17 to 2024-04-21. Dataset: Google/EDNS-CS**

## 4.3 How Quickly Do Top Website's Front-ends Shift?

We run Fenrir on two top websites: Google's front page and Wikipedia's home page. Our analysis shows Google employs aggressive deployment [23] to frequently change routing; on the other hand, Wikipedia's clients tend to go to the same site.

*4.3.1 Google.* Google is certainly a top-ten website, so we study them as an example of how such websites behave. Google began two policies that have been widely emulated: first, they aggressively deployed thousands of front ends around the Internet [11], so they could place their content near end-users, often directly peering with "eyeball" networks and enterprises (as we saw in §4.1). Second, Google frequently updates production services, and migrates customers between front ends, so this churn is not generally visible [23].

We study how Google presents itself to customers by looking at their front web page and using the EDNS/Client-Subnet extension to see what server is assigned to global networks. These servers are the Google customer catchments.

*Change in Routing Modes:* Figure 5 shows the heatmap of routing changes of Google's front-ends. We have data for two discontiguous periods, three days starting 2013-05-26 [11], and then 60 days starting 2024-02-21. The heatmap shows the full range of routing modes similarity, and the top three lines are eight years before the remainder of the graph.

The overall trend shows a fairly strong mode, with strong similarities lasting about seven days and significant differences with the next period. Φ within each week are pretty high, around 0.79, while between periods Φ are only around 0.25. That mode suggests regularly scheduled changes corresponding with the work week. Some sequences of periods show larger similarities, such as the periods starting 2024-03-03, and then -11 and -18, and the three periods starting 2024-03-24, -04-01, and -04-07.

As one would expect, data from 2013 is quite different from today. The three days starting 2013-05-26 have no similarity with any of the modern infrastructure—Google has completely changed its front-end infrastructures after ten years.

Overall, these results show Google's policy of aggressive deployment [23] means clients often change the front ends they are using.

*4.3.2 Wikipeida.* Wikipedia is another top-10 website. Wikipedia is a non-profit and operates seven Front-end sites, each serving clients around the world, with clients associated with site based on client location. With a much smaller footprint and different site-slection method, Wikpedia complements our examination of Google.

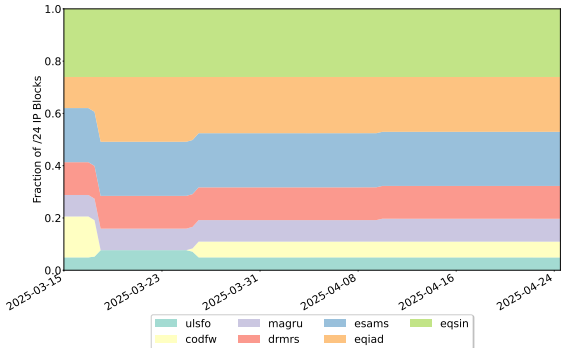
We measure Wikipedia following §2.3.3 and show Wikipedia catchments in Figure 6.

*Change in Routing Modes:* We see that Wikipedia's catchments are very stable over this observation period, with each mode showing $\Phi$ in [0.93, 0.95].

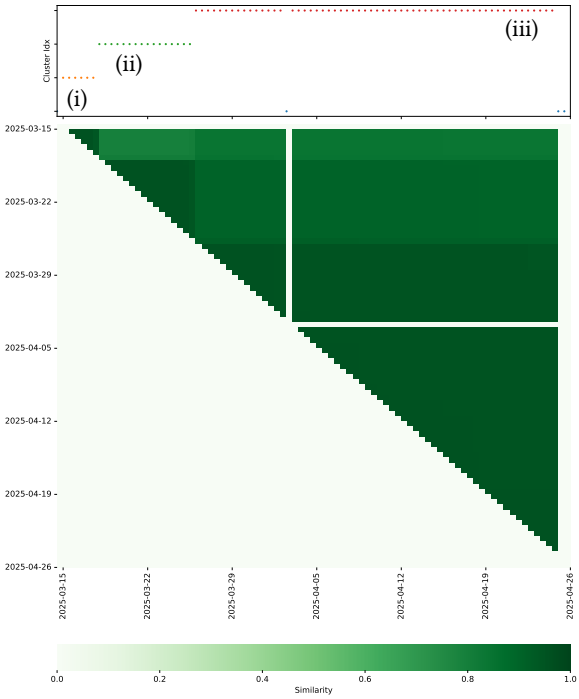
The exception is mode (ii), the week starting 2025-03-19. Fenrir shows $\Phi(M_i, M_{ii})$ = [0.79.0.94], so about 20% of networks shift when one site goes offline. Figure 6a shows the absence of codfs, with 75% of its clients going to eqiad and 25% to ulsfo.

We also see that when codfw returns on 2025-03-26, the new mode is slightly different from before: $\Phi(M_i, M_{iii})$ = [0.8.0.94], since only 30% of codfw's original client return.

The Wikipedia operators knew about codfw going offline. This event is confirmed on Wikipedia's public dashboard [2], indicating codfw was drained on 2025-03-19, and was back online on 2025-03-26. Fenrir helps quantify how different the post-event catchments are compared to pre-event. For websites that use anycast instead of geographic catchment selection, Fenrir also helps discover catchment changes that result from third-party routing changes.




**(a) The aggregated catchment distribution of Wikipedia.**




**(b) Heatmap comparing vectors of Wikipedia.**

**Figure 6: Wikipedia catchments from 2025-03-15 to 2025-04-26. Dataset: Wiki/EDNS-CS.**

## 5 RELATED WORK

Our work builds on prior work studying routing and anycast to optimize routing.

**Routing optimization and measurement for enterprises and hypergiants:** A number of commercial tools optimize routing for enterprises [44] and to provide general reporting [59]. Hypergiants optimize their internal WANs [28] and their IXP- and customer-facing links [5, 28, 51, 63]. Often this prior work depends on global knowledge of ISP or hypergiant traffic and detailed access to the control plan (routing) and data plane (traffic) information. We share these goals, but focus on observing changes with unprivileged tools (DNS and traceroutes).

A great deal of routing analysis considers control-plane measurements from BGP peering sessions. Very close to our work is work from Boston University uses hierarchical clustering and similarity heatmaps to study routing to Google, an anycast DNS, and an IXP [17]. Unlike our work, their goal is to identify "unusual routes" and hijacks in control-plane data, We currently use data-plane sources to detect routing stability in several ways. Others study BGP hijacking with RouteViews (control-plane) data [54], to protect applications [58], or to evaluate risks to specific applications like Tor [57]. While in principle, our approach could use control-plane information as a data source, demonstrating that it is future work. We share the goal of these control-plane tools

to understand routing, but they often go deeper into specific threats (hijacking) or applications (Tor).

RouteViews [60] and RIPE RIS [45] are a long-term BGP data repository, and several general control-plane analysis tools have been deployed [13, 40] using it or similar information. These tools have been applied to track Internet growth [38] and structure [39]. We instead use data-plane information to evaluate how routing changes affect specific services.

**Anycast measurement and optimization:** IP anycast [42] is widely used today by DNS [16, 25] and CDNs [29, 53] to minimize latency [52] and manage DDoS [35, 49]. Users of IP anycast have developed a special tools to evaluate destinations with active probing [18, 56, 62] and passive analysis [10, 36]. They have applied these tools to identify and resolve specific problems like anycast polarization [36, 50], suboptimal routing [9, 61] and to evaluate specific applications like regional anycast systems [65] and enterprise ingress traffic [30]. These approaches identify specific performance questions and optimize anycast networks in different ways. Our work instead focuses broadly on anycast situational awareness, and could serve as a trigger to prompt using the above approaches to investigate current user performance or opportunities to optimize anycast.

**Measuring web services:** There is a rich field in measuring web services, with both commercial approaches (for example, [59]) and research approaches. Our work instead focuses on routing, not web performance specifically. However, we benefit from the EDNS/client-subnet methods begun to study Google [11].

## 6 CONCLUSION

This paper presented present Fenrir, a tool to evaluate recurring routing results, with applications to anycast services, multi-homed networks, and top websites deployed across many front-ends. We describe how Fenrir can ingest data from active probing taken from several different measurement tools such as RIPE Atlas, traceroutes, and website lookups. After data cleaning, Fenrir reduces this data to a *routing vector* that documents how user networks reach the service of interest, defining *catchments* that can optionally be weighted to reflect service performance. We then describe three techniques for routing vector analysis: pairwise comparison, hierarchical clustering, and visualization through all-pairs heatmaps and vector transition matrices.

We apply Fenrir to evaluate routing in three specific applications: B-Root as an example of a service using IP anycast, routing out of a multi-homed enterprise, and google as an example website served by an extensive network of front-ends. We leverage existing datasets spanning five years, and collect customized new monthly datasets to study these systems. We validate our approach for B-Root anycast, using operator-provided ground truth, showing perfect recall (1) and good accuracy (0.84). Precision is lower (0.7), but because Fenrir's goal is to detect not only operator-known events, but also to detect *third-party* changes to routing of which operators would otherwise be unaware.

Applying our tool to these applications, clustering identifies several common "modes" in B-Root anycast, discovering that about one-third of catchments between 2019 and 2024 match. Based on eight months of data of the multi-homed enterprise USC, we show that its routing is very stable at its immediate upstreams; however, a huge routing change results in at most 90% of catchment changes. Finally, examination of Google's homepage as an example of top website captures its aggressive policy for rapid service evolution, particularly compared to our other subjects. Users see very similar front-ends (79% similar) for a week, but only 25% similar across weeks. On the other hand, Wikipedia's routing is relatively stable, except the time when one site was down; after the site was up again, the new routing result is only 80% similar to the previous one. These several applications show that Fenrir is a useful general tool to understand Internet-wide routing.

**Acknowledgments:** This work was partially supported by NSF through grants CNS-2212480 and CNS-2319409.

# REFERENCES


[1] [n. d.]. Global traffic routing - Wikitech — wikitech.wikimedia.org. https://wikitech.wikimedia.org/wiki/Global_traffic_routing. ([n. d.]). [Accessed 10-05-2025].

[2] [n. d.]. Grafana — grafana.wikimedia.org. https://grafana.wikimedia.org/d/000000093/cdn-frontend-network. ([n. d.]). [Accessed 10-05-2025].

[3] Emile Aben. 2024. Does the Internet Route Around Damage? Baltic Sea Cable Cuts. RIPE Labs Blog https://labs.ripe.net/author/emileaben/does-the-internet-route-around-damage-baltic-sea-cable-cuts/. (Nov. 2024). https://labs.ripe.net/author/emileaben/does-the-internet-route-around-damage-baltic-sea-cable-cuts/

[4] J. Abley and K. Lindqvist. 2006. *Operation of Anycast Services.* RFC 4786. Internet Request For Comments. https://doi.org/10.17487/RFC4786 (also Internet BCP-126).

[5] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. 2012. Anatomy of a Large European IXP. In *Proceedings of the ACM SIGCOMM Conference.* ACM, Helsinki, Finland, 163–174. http://conferences.sigcomm.org/sigcomm/2012/paper/sigcomm/p163.pdf

[6] Hussein A. Alzoubi, Seungjoon Lee, Oliver Spatscheck, Michael Rabinovich, and Jacobus Van der Merwe. 2008. Anycast CDNs Revisited. In *Proceedings of the International Conference on World Wide Web.* ACM, Beijing, China, 277–286. https://doi.org/10.1145/1367497.1367536

[7] B-Root. 2023. B-Root's anycast instance in Chile is moving. (May 2023). https://b.root-servers.org/news/2023/05/05/chile-anycast.html

[8] Ray Bellis. 2010. F-Root Routing: How Does It Work? ISC blog https://www.isc.org/blogs/f-root-routing-how-does-it-work/. (Aug. 2010). https://www.isc.org/blogs/f-root-routing-how-does-it-work/

[9] Ray Bellis. 2015. Researching F-root Anycast Placement Using RIPE Atlas. ripe blog https://labs.ripe.net/Members/ray_bellis/researching-f-root-anycast-placement-using-ripe-atlas. (Oct. 2015). https://labs.ripe.net/Members/ray_bellis/researching-f-root-anycast-placement-using-ripe-atlas

[10] Rui Bian, Shuai Hao, Haining Wang, Amogh Dhamdere, Alberto Dainotti, and Chase Cotton. 2019. Towards passive analysis of anycast in global routing: unintended impact of remote peering. *ACM Computer Communication Review* 49, 3 (Nov. 2019). https://doi.org/10.1145/3371927.3371930

[11] Matt Calder, Xun Fan, Zi Hu, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. 2013. Mapping the Expansion of Google's Serving Infrastructure. In *Proceedings of the ACM Internet Measurement Conference.* ACM, Barcelona, Spain, 313–326. https://doi.org/10.1145/2504730.2504754

[12] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. 2015. Analyzing the Performance of an Anycast CDN. In *Proceedings of the ACM Internet Measurement Conference.* ACM, Tokyo, Japan, 531–537. https://doi.org/10.1145/2815675.2815717

[13] Ying-Ju Chi, Ricardo Oliveira, and Lixia Zhang. 2008. Cyclops: the AS-level connectivity observatory. *ACM Computer Communication Review* 38, 5 (Oct. 2008), 5–16. https://doi.org/10.1145/1452335.1452337

[14] Yi-Ching Chiu, Brandon Schlinker, Abhishek Balaji Radhakrishnan, Ethan Katz-Bassett, and Ramesh Govindan. 2015. Are We One Hop Away from a Better Internet?. In *Proceedings of the ACM Internet Measurement Conference.* ACM, Tokyo, Japan, 523–529. https://doi.org/10.1145/2815675.2815719.

[15] David D. Clark. 1988. The Design Philosophy of the DARPA Internet Protocols. In *Proceedings of the 1988 Symposium on Communications Architectures and Protocols.* ACM, 106–114. https://doi.org/10.1145/205447.205458

[16] Lorenzo Colitti. 2006. K-root anycast deployment. Presentation at RIPE-52. (April 2006). http://meetings.ripe.net/ripe-52/presentations/ripe52-dns-kroot-anycast.pdf

[17] Giovanni Comarela, Evimaria Terzi, and Mark Crovella. 2016. Detecting Unusually-Routed ASes: Methods and Applications. In *Proceedings of the ACM Internet Measurement Conference.* ACM, Santa Monica, CA, USA. https://doi.org/10.1145/2987443.2987478

[18] Wouter B. de Vries, Ricardo de O. Schmidt, Wes Hardaker, John Heidemann, Pieter-Tjerk de Boer, and Aiko Pras. 2017. Verfploeter: Broad and Load-Aware Anycast Mapping. In *Proceedings of the ACM Internet Measurement Conference.* ACM, London, UK. https://doi.org/10.1145/3131365.3131371

[19] Xun Fan, John Heidemann, and Ramesh Govindan. 2013. Evaluating Anycast in the Domain Name System. In *Proceedings of the IEEE Infocom.* IEEE, Turin, Italy, 1681–1689. https://doi.org/10.1109/INFCOM.2013.6566965

[20] Xun Fan, Ethan Katz-Bassett, and John Heidemann. 2015. Assessing Affinity Between Users and CDN Sites. In *Proceedings of the 7th IEEE International Workshop on Traffic Monitoring and Analysis.* IEEE, Barcelona, Spain, 95–110. https://doi.org/10.1007/978-3-319-17172-2_7

[21] Ashley Flavel, Pradeepkumar Mani, David A. Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. 2015. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *Proceedings of the USENIX Symposium on Network Systems Design and Implementation.* USENIX, Oakland, CA, USA. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/flavel

[22] Romain Fontugne, Anant Shah, and Emile Aben. 2018. The (Thin) Bridges of AS Connectivity: Measuring Dependency Using AS Hegemony. In *Proceedings of the Passive and Active Measurement Conference.* Springer, Berlin, Germany, 216–227. https://doi.org/10.1007/978-3-319-76481-8_16

[23] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2016. Evolve or Die: High-Availability Design Principles Drawn from Google's Network Infrastructure. In *Proceedings of the ACM SIGCOMM Conference.* ACM, Florianopolis, Brazil, 58–72. https://doi.org/10.1145/2934872.2934891

[24] John Clifford Gower. 1985. Properties of Euclidean and non-Euclidean distance matrices. *Linear algebra and its applications* 67 (1985), 81–97.

[25] T. Hardie. 2002. *Distributing Authoritative Name Servers via Shared Unicast Addresses.* RFC 3258. Internet Request For Comments. https://www.rfc-editor.org/rfc/rfc3258.txt (informational).

[26] Keqiang He, Alexis Fisher, Liang Wang, Aaron Gember, Aditya Akella, and Thomas Ristenpart. 2013. Next Stop, the Cloud: Understanding Modern Web Service Deployment in EC2 and Azure. In *Proceedings of the ACM Internet Measurement Conference.* ACM, Barcelona, Spain, 177–190. http://conferences.sigcomm.org/imc/2013/papers/imc097-heA.pdf

[27] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. 2008. Census and survey of the visible internet. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC '08).* Association for Computing Machinery, New York, NY, USA, 169–182. https://doi.org/10.1145/1452520.1452542

[28] Sushant Jain, Alok Kumar, Joon Ong Subhasree Mandal, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Jonathan Zolla Min Zhu, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2013. B4: Experience with a Globally-Deployed Software Defined WAN. In *Proceedings of the ACM SIGCOMM Conference.* ACM, Hong Kong, China, 3–14. https://doi.org/10.1145/2534169.2486019

[29] Thomas Koch, Ke Li, Calvin Ardi, Ethan Katz-Bassett, Matt Calder, and John Heidemann. 2021. Anycast in Context: A Tale of Two Systems. In *Proceedings of the ACM SIGCOMM Conference.* ACM, Virtual. https:

//doi.org/10.1145/3452296.3472891

[30] Thomas Koch, Shuyue Yu, Sharad Agarwal, Ethan Katz-Bassett, and Ryan Beckett. 2023. PAINTER: Ingress Traffic Engineering and Routing for Enterprise Cloud Networks. In *Proceedings of the ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 360–377. https://doi.org/10.1145/3603269.3604868

[31] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. 2010. Internet Inter-Domain Traffic. In *Proceedings of the ACM SIGCOMM Conference*. ACM, New Delhi, India, 75–86. https://doi.org/10.1145/1851182.1851194

[32] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2018. Internet Anycast: Performance, Problems, and Potential. In *Proceedings of the ACM SIGCOMM Conference*. ACM, Budapest, Hungary, 59–73. https://doi.org/10.1145/3230543.3230547

[33] Ziqian Liu, Bradley Huffaker, Marina Fomenkov, Nevil Brownlee, and kc claffy. 2007. Two Days in the Life of the DNS Anycast Root Servers. In *Proceedings of the Passive and Active Measurement Conference*. Springer-Verlag, Louvain-la-neuve, Belgium, 125–134. https://www.caida.org/publications/papers/2007/dns_anycast/dns_anycast.pdf

[34] M Luckie. [n. d.]. Scamper: a scalable and extensible packet prober for active measurement of the internet. https://catalog.caida.org/paper/2010_scamper. ([n. d.]). https://doi.org/paper/2010_scamper Accessed: <date accessed>.

[35] Giovane C. M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Müller, Lan Wei, and Christian Hesselman. 2016. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proceedings of the ACM Internet Measurement Conference*. ACM, Santa Monica, California, USA, 255–270. https://doi.org/10.1145/2987443.2987446

[36] Giovane C. M. Moura, John Heidemann, Wes Hardaker, Pithayuth Charnsethikul, Jeroen Bulten, João M. Ceron, and Cristian Hesselman. 2022. Old but Gold: Prospecting TCP to Engineer and Live Monitor DNS Anycast. In *Proceedings of the Passive and Active Measurement Conference*. Springer, virtual, 264–292. https://doi.org/10.1007/978-3-030-98785-5_12 Awarded best paper.

[37] Giovane C. M. Moura, John Heidemann, Moritz Müller, Ricardo de O. Schmidt, and Marco Davids. 2018. When the Dike Breaks: Dissecting DNS Defenses During DDoS. In *Proceedings of the ACM Internet Measurement Conference*. ACM, Boston, MA, USA, 8–21. https://doi.org/10.1145/3278532.3278534

[38] Ricardo Oliveira, Beichuan Zhang, and Lixia Zhang. 2007. Observing the Evolution of Internet AS Topology. In *Proceedings of the ACM SIGCOMM Conference*. ACM, Kyoto, Japan, 313–324. https://doi.org/10.1145/1282427.1282416

[39] Ricardo V. Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. 2008. In search of the elusive ground truth: the Internet's AS-level connectivity structure. In *Proceedings of the ACM SIGMETRICS*. ACM, 217–228. https://doi.org/10.1145/1384529.1375482

[40] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. 2016. BGPStream: A Software Framework for Live and Historical BGP Data Analysis. In *Proceedings of the ACM Internet Measurement Conference*. ACM, Santa Monica, CA, USA. https://doi.org/10.1145/2987443.2987482

[41] Bojan Pancevski. 2024. Chinese Ship's Crew Suspected of Deliberately Dragging Anchor for 100 Miles to Cut Baltic Cables. Wall Street Journal. (Nov. 2024). https://www.msn.com/en-us/news/world/chinese-ship-suspected-of-deliberately-dragging-anchor-for-100-miles-to-cut-baltic-cables/ar-AA1uRlcC

[42] C. Partridge, T. Mendez, and W. Milliken. 1993. *Host Anycasting Service*. RFC 1546. Internet Request For Comments. https://www.rfc-editor.org/rfc/rfc1546.txt

[43] Lin Quan, John Heidemann, and Yuri Pradkin. 2013. Trinocular: understanding internet reliability through adaptive probing. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. Association for Computing Machinery, New York, NY, USA, 255–266. https://doi.org/10.1145/2486001.2486017

[44] Jennifer Rexford. 2006. *Handbook of Optimization in Telecommunications*. Springer, Chapter Route Optimization in IP Networks, 679–700. https://doi.org/10.1007/978-0-387-30165-5_24

[45] RIPE. 2011. Routing Information Service (RIS). web site http://www.ripe.net/data-tools/stats/ris. (2011). http://www.ripe.net/data-tools/stats/ris

[46] RIPE NCC. 2015. DNSMON. web site https://atlas.ripe.net/dnsmon/. (2015). https://atlas.ripe.net/dnsmon/

[47] RIPE NCC Labs. 2023. *Internet Country Report: Türkiye*. Technical Report. RIPE. https://labs.ripe.net/media/documents/RIPE_NCC_Country_Report_2023_Turkiye.pdf

[48] RIPE NCC Staff. 2015. RIPE Atlas: A Global Internet Measurement Network. *The Internet Protocol Journal* 18, 3 (Sept. 2015), 2–26. http://ipj.dreamhosters.com/wp-content/uploads/2015/10/ipj18.3.pdf

[49] A S M Rizvi, Leandro Bertholdo, João Ceron, and John Heidemann. 2022. Anycast Agility: Network Playbooks to Fight DDoS. In *Proceedings of the 31st USENIX Security Symposium*. USENIX, 4201–4218.

[50] ASM Rizvi, Tingshan Huang, Rasit Esrefoglu, and John Heidemann. 2024. Anycast Polarization in The Wild. In *Proceedings of the Passive and Active Measurement Conference*. Springer, Virtual Location.

[51] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V. Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. 2017. Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. In *Proceedings of the ACM SIGCOMM Conference*. ACM, Los Angeles, CA, USA, 418–431. https://doi.org/10.1145/3098822.3098853

[52] Ricardo de O. Schmidt, John Heidemann, and Jan Harm Kuipers. 2017. Anycast Latency: How Many Sites Are Enough?. In *Proceedings of the Passive and Active Measurement Conference*. Springer, Sydney, Australia, 188–200.

[53] Kyle Schomp, Onkar Bhardwaj, Eymen Kurdoglu, Mashooq Muhaimen, and Ramesh K. Sitaraman. 2020. Akamai DNS: Providing Authoritative Answers to the World's Queries. In *Proceedings of the ACM SIGCOMM Conference*. ACM, Virtual, 465–478. https://doi.org/10.1145/3387514.3405881

[54] Anant Shah, Romain Fontugne, and Christos Papadopoulos. 2016. Towards Characterizing International Routing Detours. In *Proceedings of the 12th Asian Internet Engineering Conference (AINTEC)*. ACM, Bangkok, Thailand. http://www.iij-ii.co.jp/en/member/romain/pdf/anant_aintec2016.pdf

[55] R. Sibson. 1973. SLINK: an optimally efficient algorithm for the single-link cluster method. *Comput. J.* 16, 1 (Jan. 1973), 30–34. https://doi.org/10.1093/comjnl/16.1.30

[56] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, van Rijswijk-Deij, Roland, Alberto Dainotti, KC Claffy, and Anna Sperotto. 2020. MAnycast2—Using Anycast to Measure Anycast. In *Proceedings of the ACM Internet Measurement Conference*. ACM, Pittsburgh, PA, USA. https://doi.org/10.1145/3419394.3423646

[57] Yixin Sun, Maria Apostolaki, Henry Birge-Lee, Laurent Vanbever, Jennifer Rexford, Mung Chiang, and Prateek Mittal. 2020. *Securing Internet Applications from Routing Attacks*. Technical Report arXiv:2004.09063v3. arXiv. https://arxiv.org/abs/2004.09063

[58] Yixin Sun, Maria Apostolaki, Henry Birge-Lee, Laurent Vanbever, Jennifer Rexford, Mung Chiang, and Prateek Mittal. 2021. Securing Internet Applications from Routing Attacks. *Commun. ACM* 64, 6 (June 2021), 86–96. https://doi.org/10.1145/3429775

[59] ThousandEyes. 2021. Internet Insights. product brief https://marketo-web.thousandeyes.com/rs/772-KGG-249/images/ThousandEyes-Product-Brief-Internet-Insights.pdf. (Nov. 2021). https://marketo-web.thousandeyes.com/rs/772-KGG-249/images/ThousandEyes-Product-Brief-Internet-Insights.pdf

[60] Route Views. 2000. University of Oregon Route Views Project. web site http://www.routeviews.org. (2000). http://www.routeviews.org

[61] Lan Wei, Marcel Flores, Harkeerat Bedi, and John Heidemann. 2020. Bidirectional Anycast/Unicast Probing (BAUP): Optimizing CDN Anycast. In *Proceedings of the IEEE Network Traffic Monitoring and Analysis Conference*. IFIP, Berlin, Germany.

[62] S. Woolf and D. Conrad. 2007. *Requirements for a Mechanism Identifying a Name Server Instance*. RFC 4892. Internet Request For Comments. https://www.rfc-editor.org/rfc/rfc2414.txt

[63] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, Victor Lin, Colin Rice, Brian Rogan, Arjun Singh, Bert Tanaka, Manish Verma, Puneet Sood, Mukarram Tariq, Matt Tierney, Dzevad Trumic, Vytautas Valancius, Calvin Ying, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2017. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *Proceedings of the ACM SIGCOMM Conference*. ACM, Los Angeles, CA, USA, 432–445. https://doi.org/10.1145/3098822.3098854

[64] Xiao Zhang, Tanmoy Sen, Zheyuan Zhang, Tim April, Balakrishnan Chandrasekaran, David Choffnes, Bruce M. Maggs, Haiying Shen, Ramesh K. Sitaraman, and Xiaowei Yang. 2021. AnyOpt: Predicting and Optimizing IP Anycast Performance. In *Proceedings of the ACM SIGCOMM Conference*. ACM. https://doi.org/10.1145/3452296.3472935

[65] Minyuan Zhou, Xiao Zhang, Shuai Hao, Xiaowei Yang, Jiaqi Zheng, Guihai Chen, and Wanchun Dou. 2023. Regional IP Anycast: Deployments, Performance, and Potentials. *ACM SIGCOMM Conference* (Sept. 2023), 917–931. https://doi.org/10.1145/3603269.3604846

# A CHANGES IN UPSTREAMS PROVIDERS OF THE ENTERPRISE NETWORK

We use the Sankey diagram to study how the flow has changed starting from the enterprise network, Figure 7 and Figure 8 illustrate the number of networks served by each upstream provider at hop 1-4. The number labeled at each node is the AS number of the upstream provider.

At hop 2 (as labeled _1 in the graph), on 2025-01-14, 8% destination networks were routed by AS 2152; on 2025-01-20, the percentage dropped to 1.5% after the huge routing change. At hop 3 (as labeled _2 in the graph), on 2025-01-14, 80% destination networks were routed by AS 2152; on 2025-01-20, the percentage dropped to 13% after the huge routing change. Also, those changed networks were instead routed by AS 2914 (31%), AS 6939 (29%), and AS 226 (22%). At hop 4 (as labeled _1 in the graph), the change in routing result is more significant, as it is further from the enterprise.

By using Sankey diagram to visualize the routing result, we can determine what upstream providers are influenced when routing changes.

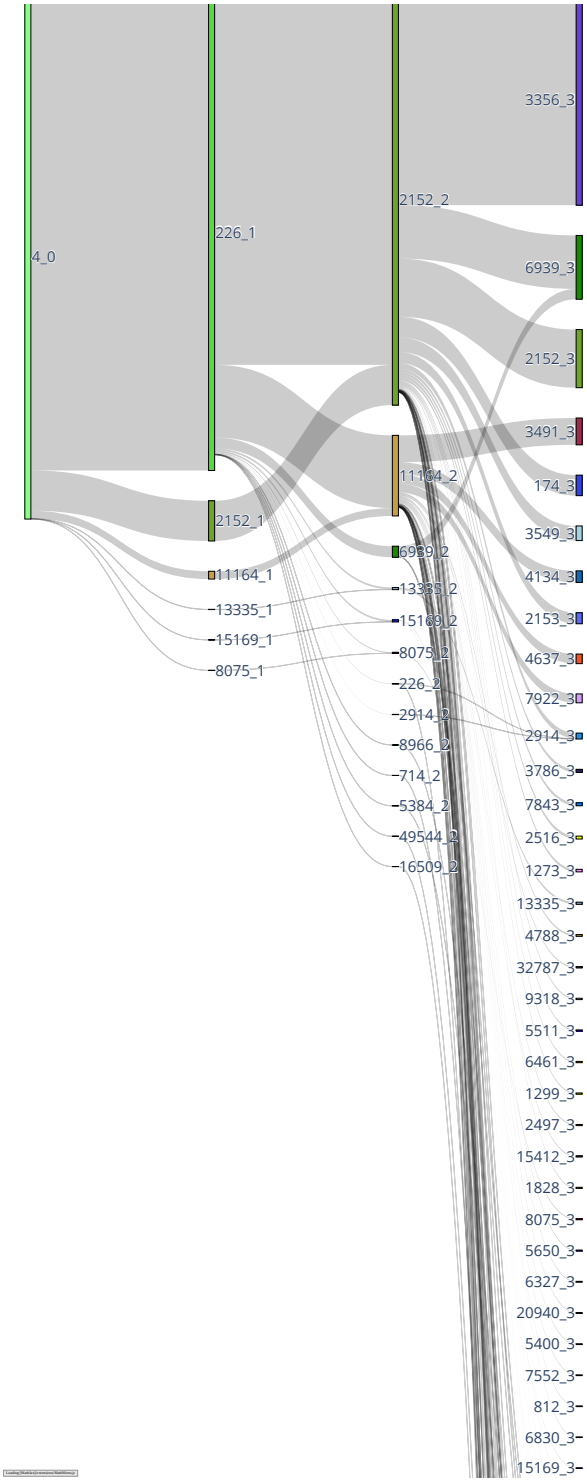



**Figure 7: The flow topology of a multi-homed enterprises before the routing change, 2025-01-14.**

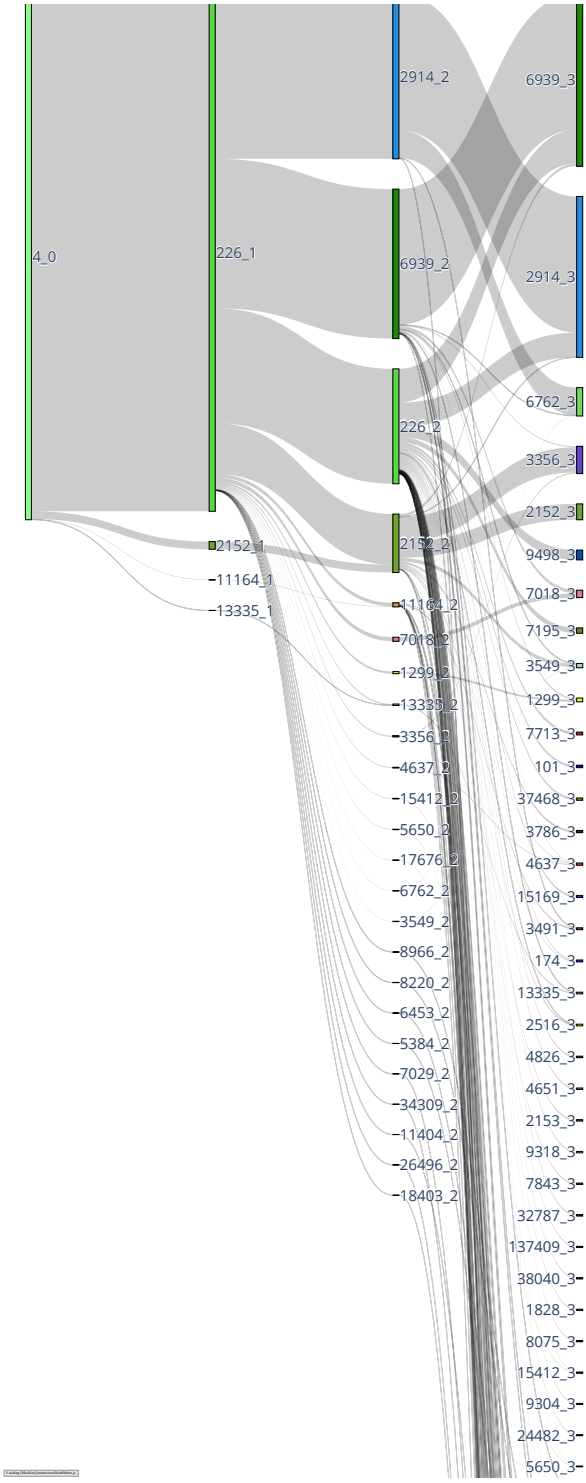



**Figure 8: The flow topology of a multi-homed enterprises after the routing change, 2025-01-20.**