The Best of Both Worlds

High Availability CDN Routing Without Compromising Control

Jiangchen Zhu
Columbia University

Kevin Vermeulen
LAAS-CNRS

Italo Cunha
Universidade Federal de Minas Gerais

Ethan Katz-Bassett
Columbia University

Matt Calder
Columbia University

ABSTRACT

Content delivery networks (CDNs) provide fast service to clients by replicating content at geographically distributed sites. Most CDNs route clients to a particular site using anycast or unicast with DNS-based redirection. We analyze anycast and unicast and explain why neither of them provides both precise control of user-to-site mapping and high availability in the face of failures, two fundamental goals of CDNs. Anycast compromises control (and hence performance), and unicast compromises availability. We then present new hybrid techniques and demonstrate via experiments on the real Internet that these techniques provide both a high level of traffic control and fast failover following site failures.

CCS CONCEPTS

• **Networks** → **Network measurement**.

KEYWORDS

Anycast, unicast, routing, performance, availability, CDN.

ACM Reference Format:
Jiangchen Zhu, Kevin Vermeulen, Italo Cunha, Ethan Katz-Bassett, and Matt Calder. 2022. The Best of Both Worlds: High Availability CDN Routing Without Compromising Control. In *ACM Internet Measurement Conference (IMC '22), October 25–27, 2022, Nice, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3517745.3561421

1 INTRODUCTION

A Content Delivery Network (CDN) provides high performance content distribution via geographically distributed sites. CDNs distribute load and provide low latency by directing clients to nearby sites. However, site failures in a distributed networked system like a CDN are common and can prevent clients from accessing content or services until they are mitigated. Site failures can inevitably originate from server software or hardware upgrades [17], hardware failures (e.g., router, line-card), network misconfiguration (e.g., BGP, DNS), facility outages [15], or sudden bursts of traffic (e.g., DDoS). CDNs need techniques that reliably direct clients to sites that provide high performance access to services and content and that quickly divert clients to alternate sites in cases of failures.

The most common current techniques to direct clients to sites for latency-sensitive services, IP unicast and IP anycast (§2), force CDNs to make a tradeoff between traffic control and availability. With IP unicast, a CDN directs users to a particular site by using DNS to return IP addresses specific to the site, but the speed at which the CDN can move users between sites is inherently limited by caching of DNS records. A DNS record's time-to-live (TTL) specifies how long it should be cached but cannot guarantee fast failover. Setting TTL too low introduces additional latency for applications, and some client software and DNS resolvers continue to use a DNS record after the TTL expires, directing traffic to the corresponding site [1, 21, 29]. IP anycast relies on BGP's distributed path selection to direct clients to sites and so lacks unicast's control [26, 27] but enables fast failover following failures by withdrawing announcements of the anycast address from the failed site [3].

To address the current need to compromise either control or availability, we present new techniques that combine the strengths of both unicast and anycast to achieve both precise traffic control and fast site failover. Our approaches rely on each site being assigned a distinct prefix (*e.g.*,/24 or /48), from which DNS records are returned as in unicast. To achieve fast site failover, in addition to updating DNS records, we demonstrate that other sites can provide alternative routes to the failing site's prefix, which is similar to IP anycast. To prevent these alternative routes from interfering with control under normal operations, the other sites either announce them only upon failure or announce them in ways that make them less preferred (*e.g.,* prepending). In this way, even if clients do not receive new DNS records immediately, packets destined to the failing site's address will instead be routed to other sites.

We evaluate our techniques with Peering [34], a testbed that lets us make BGP announcements and exchange traffic with the real Internet at different sites. We show that our techniques combine traffic control and fast site failover better than existing techniques.

- Our first technique (§4) is able to retain the same amount of traffic control as unicast and has a failover time of 10 seconds in the median (for hosts across the Internet and emulated failures of each Peering site), only 2 seconds longer than IP anycast. In contrast, the DNS TTLs used by top domains are around 10 minutes at median [29]. Although some CDNs use much lower DNS TTLs (*e.g.,* Akamai uses 20s [37]), some clients continue using out-of-date DNS records in violation of TTL [1].
- Compared to our first technique, our second technique (§4) provides even faster failover, at the cost of some control. Compared to anycast, it achieves roughly the same failover time, but, of the

many clients that anycast directs to a suboptimal site [27], our technique is able to correctly steer 60% of them.

We compare the routing convergence properties of PEERING announcements with those of real-world hypergiant networks and conclude that the results can be generalized to real CDNs.

2 BACKGROUND

Unicast-based CDN site selection. In DNS-based redirection [9], each site announces a unique unicast prefix. The CDN's authoritative DNS resolver returns an IP address within the optimal site's prefix (*e.g.,* based on performance and load). We call this technique *unicast* in the rest of the paper to highlight its announcement strategy, as all techniques use DNS to provide IP addresses to clients.

During failures, the CDN updates DNS records and relies on the clients requerying DNS for IP addresses to other sites, but records are cached by the clients' recursive resolvers, OSes, and, potentially, applications. The CDN specifies a time-to-live (TTL) value in the DNS record that indicates the maximum time the record should be cached. Although the TTL can be set to a small value (*e.g.,* less than 60s), this increases application latency [4, 40], and some applications and recursive resolvers violate the TTL. A recent study found that many connections were established after the TTL expired, and the median time since expiration was 890s [1].

Anycast-based CDN site selection. With anycast, each site announces the same IP prefix, and BGP policy routes clients to a particular site. Since the path taken to the CDN's network is determined by the BGP policy of other networks, the CDN does not have direct traffic control for performance and load management. Previous work showed that a subset of clients are routed to suboptimal sites [7, 27]. Despite this, IP anycast has some advantages during site failure: no DNS records need updating, and, if a site fails, it can withdraw its announcements, and the remaining anycast announcements from other sites will quickly attract traffic from clients that previously had gone to the failed site. Although the failover time is subject to BGP convergence, previous work [3] and our work show that IP anycast achieves failover in tens of seconds.

3 GOALS AND NON-SOLUTIONS

Goals. Our goal is to develop techniques that overcome the limitations of current redirection techniques (§2): achieve both the control over client-to-site mapping of unicast and the availability in the face of site failure of anycast. Control is necessary because only the CDN has access to the service availability, server load, and internal software and hardware health information necessary to make the best redirection decisions. High availability is critical to a CDN's business and is closely tied to its ability to quickly direct clients away from a failed (or failing) site.

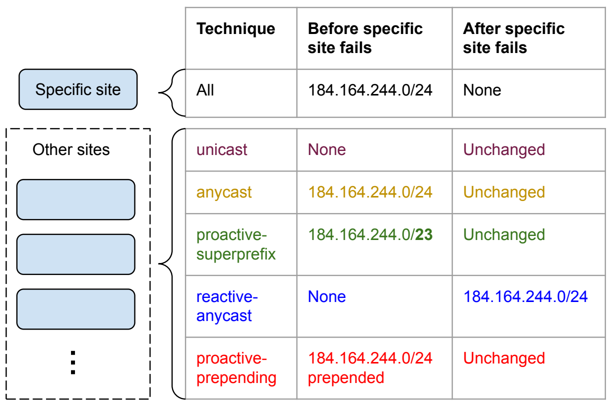
Hybrid non-solutions. Our goal of combining some of the advantages of unicast with the advantages of anycast suggests the use of hybrid solutions. While the approaches we propose in Section 4 are in fact hybrid, the two existing hybrid approaches of which we are aware are not good solutions to our goal.

First, our prior work proposed identifying the subset of clients with poor anycast performance and using unicast just for these clients [7]. While this approach can overcome the disadvantages of anycast for clients of the CDN's choice, it comes at the expense of inheriting the disadvantages of unicast for that subset of clients.

Second, a CDN can announce a unicast prefix from a particular site while also announcing a less-specific covering prefix from other sites. For example, a specific site advertises 184.164.244.0/24, and all sites advertise 184.164.244.0/23 as backup (`proactive-superprefix` in Figure 1). DNS returns an address in 184.164.244.0/24 to route clients to the specific site. Because Internet routing uses longest-matching prefixes, as long as routes for the unicast prefix exist, this approach enjoys the same traffic control as unicast, but, once the unicast prefix is withdrawn following a failure, traffic to addresses in that prefix will instead use routes to the covering prefix announced from other sites, even before receiving a new DNS record. We described this approach, which we call `proactive-superprefix`, to colleagues at three major CDNs, and all thought it would improve failover and availability for a CDN using unicast. To find evidence of possible use, we examined a RIS [30] BGP dump for prefixes containing IP addresses of hypergiant web servers from April 2021 [14]. Of the most specific prefixes that hosted these servers and were announced by the hypergiants, 39% were also covered by less specific prefixes announced by the hypergiants at the same time, with the value ranging from 12% to 95% for individual hypergiants.

Despite the simplicity and great traffic control, the failover time is poor. After a failed site withdraws its unicast prefix, (invalid) routes to the prefix still exist in routers during route convergence, which is slow for unicast prefixes [24] due to the lack of valid competing routes for the exact prefix from other sites. When a router's preferred route to the prefix is withdrawn, if it has a route to the prefix from another neighbor, it will select one and perhaps announce it to other neighbors, but the route is necessarily invalid as the failed site withdrew its unicast announcement. BGP will continue choosing invalid routes until none exist, taking time to reach a consistent view. Until that happens, longest prefix matching means that routers will use invalid routes to the withdrawn prefix over valid routes from other sites to the covering superprefix, delaying failover. Even packets forwarded by a router that has already converged to only have the covering superprefix may encounter routers that still have invalid routes to the withdrawn prefix, preventing the packets from reaching alternate sites. In contrast, the withdrawal of an anycast prefix from a site converges faster because valid paths to alternative sites are already learned by some routers, which can quickly reconverge to these alternate routes (§5.4.1).

We study the convergence time for withdrawals of unicast prefixes both by making our own announcements from PEERING and by inspecting BGP collector archives to identify instances when a prefix that had been announced by a hypergiant is withdrawn from all collectors. We find that they take ~100s to converge at the median (across ⟨`BGP collector peer, withdrawal event`⟩) and more than 10 minutes at the tail (appendix A). Given that the median DNS TTL for popular domains is 10 minutes [29] and DNS records may be used past TTL expiration [1], this result suggests that `proactive-superprefix` may improve availability over pure unicast during some failures for some domains, but 100 seconds—much less 10 minutes—of unavailability during route convergence will quickly exhaust the unavailability budget of a CDN (*e.g.,* a few minutes per month) that hosts important Internet services [17].



Technique	Before specific site fails	After specific site fails
All	184.164.244.0/24	None
unicast	None	Unchanged
anycast	184.164.244.0/24	Unchanged
proactive-superprefix	184.164.244.0/**23**	Unchanged
reactive-anycast	None	184.164.244.0/24
proactive-prepending	184.164.244.0/24 prepended	Unchanged

Figure 1: Announcements made by the specific site and other sites before and after the specific site fails.

4 OUR TECHNIQUES

Existing techniques (§2) and hybrid non-solutions (§3) are not able to achieve site failover without compromising traffic control. This section presents two new techniques that achieve fast site failover and preserve traffic control during normal operations. For traffic control, a CDN needs the ability to redirect clients to specific sites. This requires each site to be assigned a unique prefix. While the requirement increases address usage for a pure anycast CDN, in practice anycast CDNs already often have per-site unicast prefixes as well [7]. On site failure, we assume that the site withdraws its prefix announcements. In normal operation, a CDN can either apply traffic control on all of its clients (like unicast) or use anycast on most clients but apply traffic control on a subset of clients where it wants specific control to avoid poor anycast routes, to achieve better load distribution, or to achieve other control-based goals [7].

`reactive-anycast` enables fast failover by introducing routes to alternative sites after failure. In normal operation, each specific site advertises a unicast prefix, and DNS returns IP addresses within that prefix. When the unicast prefix is withdrawn from the site due to site failure, the CDN's monitoring and control system will cause all other sites to immediately announce it (Fig. 1). The new announcements introduce new valid paths to the same prefix into the Internet, and routers can select them to replace invalid paths to the failed site, speeding up convergence. The failover time depends on how quickly the new routes propagate to and are selected by the clients or clients' upstream networks. The full propagation of new routes to all networks is not needed for failover. Immediately after the withdrawal of the unicast prefix, a client network that is not directly connected to the CDN still has an outdated path that goes through its upstream to the site, so it will still forward packets to its upstream provider. As soon as the upstream gets a new route to another site, the packets will reach the CDN, even before the client network learns of the new route.

The failover mechanism of `reactive-anycast` is similar to anycast in that both rely on other networks replacing the invalid paths with valid paths to other sites, but it differs in that this technique only introduces paths to alternate sites after failure and requires them to be propagated to clients or their upstreams. So the deciding factor for failover is how quickly an anycast announcement can propagate. We find that anycast announcements propagate fast (~10s at median across public BGP collectors) by making announcements on the PEERING testbed and looking into announcements on the real Internet (appendix B). This result makes us believe that `reactive-anycast` can failover faster than `proactive-superprefix` (§3). However, during convergence of `reactive-anycast`, some routers might lose routes, if the withdrawal reaches them before an alternate route. One might think that combining the two approaches would work better than either on its own—`proactive-superprefix` could provide a covering "backup" for any routers that the withdrawal reaches before an alternate route, and this covering route could lead packets to routers with more specific routes to alternate sites. We implemented this combined technique, and our experiments (similar to those in Section 5) revealed that it is only faster than `reactive-anycast` for the fastest 20% of failovers, and it is much worse in the long tail, an undesirable tradeoff. In future work we will investigate the dynamics more.

`reactive-anycast` maintains fine-grained traffic control because the prefix is unicast in normal operation, allowing clients to be directed to specific sites via DNS. It requires a real-time monitoring system to detect site outages, similar to ones that CDNs have deployed to quickly detect problems [5, 8]. To minimize the failover time, CDNs need to make new announcements quickly after the detection of an outage. Such real-time action has been applied in some traffic engineering systems [13, 36]. However, a disadvantage of `reactive-anycast` is that global routing configuration must be updated in response to a failure. Such simultaneous global configuration changes are operationally treacherous [33], potentially resulting in unexpected and cascading routing changes and introducing the potential for a global outage from a simple mistake [19]. To debug the propagation of the new anycast announcement, prior to failure, a CDN can rotate through its sites and withdraw a test prefix at the site to see if its clients are routed as expected.

`proactive-prepending` overcomes the shortcoming of `reactive-anycast` by introducing alternative routes ahead of failure. In normal operation, a specific site advertises a prefix without prepending its AS path, and other sites also advertise the same prefix with prepending as backup routes (Fig. 1). If the specific site goes down and withdraws its announcement, prepended routes to other sites provide reachability.

`proactive-prepending` sacrifices some traffic control compared to unicast because AS path length is not the top factor in BGP decisions. A network could prefer a prepended route due to `LOCAL_PREF`, for example if it only peers with a site making the prepended announcement. There is an interesting tradeoff. On one hand, if the other sites prepend more times, the CDN may get more traffic control because the non-prepended route to the specific site is more likely to be chosen. On the other hand, additional prepending will also make the backup routes longer than additional invalid routes following a failure, so it may take longer for them to be preferred, delaying failover. We will present results about traffic control for different prepending lengths in Section 5.4.2.

To limit the loss of control, a CDN can only announce the prepended route for a site's prefix to neighbors that also connect to the site and hence receive the non-prepended route. (It has been argued that best practice is to make consistent advertisements to a neighbor at different peering locations [12], but we are simply replacing the inconsistent advertisement of a CDN using a unicast advertisement tied to one site with anycast announcements varying path lengths. BGP MED could also be used for neighbors that support it.) For such a neighbor, the `LOCAL_PREF` is likely to be the same for all announcements from the CDN, leading all its routers to choose the non-prepended route to the intended site. If the peer sets different `LOCAL_PREF` on announcements from different sites, then the CDN will notice that all traffic goes to the sites with higher `LOCAL_PREF` and can complain to the peer. Networks also tend to set the same policy on peering points in the same continent [16]. After the site fails and withdraws the announcement, the border routers receiving prepended routes will select and then propagate them. This failover is similar to that of `reactive-anycast` but does not require global routing reconfiguration.

Announcing the prepended routes to neighbors with multiple connections to the CDN will position backup routes in two important classes of CDN neighbors. First, most CDN traffic is exchanged with a small number of eyeball networks that often serve multiple metropolitan regions within a country or region, and a CDN will typically connect to such networks in as many locations as possible. Second, CDNs often connect to the same tier-1 or large regional providers across many sites. For example, the majority of Microsoft's bandwidth costs are for traffic to 3 North American ISPs, with tens of peering points [38]. Backup routes in these two key classes provide outsized benefit to availability, as the eyeball networks host most users and the providers carry traffic between the CDN and many other networks. By prepending the routes from backup sites, the CDN retains the control necessary to take advantage of the full information it has about its customers and services [9]. In addition, with modern intent-based centralized configuration management, managing site- and/or peer-specific configurations is straightforward [34], so adding or removing a peer introduces only manageable and automatable configuration changes.

5 EVALUATION

We conduct failover measurements comparing our techniques `reactive-anycast` (§4) and `proactive-prepending` (§4) to other approaches, `proactive-superprefix` (§3) and pure IP anycast (§2). We use the Peering testbed [34] to emulate a small-scale CDN on the real Internet, emulating site failures by withdrawing announcements from one Peering site. We measure failover time on the control plane via BGP route collectors and on the data plane by issuing pings out from Peering to targets across the Internet to assess when responses reach other Peering sites. We do not perform experiments to study the failover time of unicast because our emulated CDN does not host real, popular services that clients worldwide reach via DNS, and so we have no straightforward way to measure the impact of DNS caching (and DNS TTL violations) worldwide, which is what determines the failover time. Peering has multiple sites, which have routers with BGP sessions with universities, providers, and/or IXP peers. We are allocated the prefix 184.164.244.0/23 and are allowed to advertise or withdraw it (and the two /24 prefixes within it) from any of the sites, with or without AS path prepending. Since a few Peering sites only have peers, we only use those that have at least one provider (and hence should be globally reachable and able to reach all destinations) and test their reachability with RIPE Atlas. The sites are located in Amsterdam, Athens, Boston, Atlanta, Belo Horizonte, Seattle (two sites), Salt Lake City, and Madison.

5.1 Target selection

To study traffic control before failure and availability after failure, we select different sets of client targets for each Peering site from ISI's IPv4 Hitlist [22]. The Hitlist provides one address per /24 that is likely to respond to pings, of which ~3.5M actually respond to our test pings. Probing targets from Peering provides wider coverage and more frequent measurements than platforms such as RIPE Atlas. From the responsive targets, we choose 2.8M prefixes that have web clients [20], which are more representative of CDN clients.

For each site, we select targets based on the following criteria.

- **Site proximity.** We only consider targets that are within 50ms round-trip latency (measured using a unicast announcement from the site). This is because CDNs generally serve clients from nearby sites, and Peering lacks sites in some regions.
- **Not routed to site by anycast.** In terms of traffic control, a target that is routed by anycast to the site can always be routed to that site by any of the techniques introduced in our work, so it suffices to evaluate traffic control on those that cannot be routed to the site by anycast, allowing us to limit probing overhead and to measure the additional control that a technique provides beyond what is possible with anycast. To see this, in `proactive-superprefix` and `reactive-anycast`, the most specific prefix is only announced at the site, so returning corresponding DNS records can route any target to the site. For `proactive-prepending`, this is because routes to other sites are prepended in `proactive-prepending`, and become less preferred than they are in anycast. To assess whether this criterion impacted the failover time we measured, we also picked an alternate set of targets without this criterion and found that failover times were very similar for both datasets. In the rest of the paper, we only present results using the set with this criterion.

Using the criteria above, we select 50K targets for each site. We choose the targets to spread them across ASes as evenly as possible, selecting randomly within an AS when there are multiple that meet the criteria. In total, our targets include more than 20K ASes.

5.2 Experiment setup

We iterate through each technique and, for each technique, through all sites, failing one at a time. First, we advertise the prefix(es) according to each technique's announcements before failure (Fig. 1). Since Peering providers differ by site, our evaluation of `proactive-prepending` prepends from all alternate sites, not just to neighbors that also connect to the non-prepended site as we believe a real CDN could. We prepend three times at other sites (Section 5.4.2 compares results for three vs. five, and the majority of sites do not see much change in traffic control). Then we wait for one hour to ensure convergence has been reached, and after that we test the reachability of targets. In this stage of the test, we ping the targets once and check which targets have responses

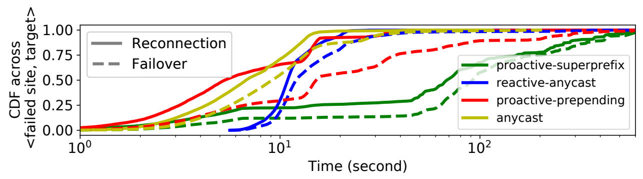


Figure 2: Reconnection and failover time for each technique. `reactive-anycast` and `proactive-prepending` achieve failover time similar to anycast. `proactive-superprefix` has a much longer failover time than anycast.

that are routed to the current site, which represent the targets controllable by the technique. Next, we emulate a site failure by withdrawing all prefixes that are announced by the current site. After the failure, we advertise prefixes from other sites if it is specified by the technique (Fig. 1), send pings to each controllable target every ~1.5s for ~600s, and run `tcpdump` at each site to record when and at which Peering site the replies from targets arrive. We use this probing to evaluate availability for our study, but real CDNs would not need this type of probing, plus already have deployed measurement solutions [8, 35]. We send the ping requests using Verfploeter [23] from a Peering site other than the failed one, using source address 184.164.244.10 so that the responses are routed towards the current site's prefix. Each ping request has a unique sequence number so that we can match each response with its request and observe whether there is a missing response (*i.e.,* disconnection). We also collect BGP feeds from RIS [30] and Routeviews [42] to show that the convergence time on Peering is similar to other networks (§5.4.3, appendix B).

5.3 Ethics

Our probing towards the targets follows established ethical measurement practices. First, these targets have not opted out during ISI's scan [11]. Second, in the payload of our ping requests, we included a link to a web page with details on our experiment and contact information to opt out. We did not receive messages to opt out. Third, we calculate that the average traffic rate during the probing period is less than 100B/s for individual targets, which is unlikely to disrupt targets or cross traffic. Fourth, we did not see evidence that we exceeded ICMP rate limits because we did not observe significant loss after convergence [18].

5.4 Results

5.4.1 Reconnection and failover time. Following a failure, a target may experience periods of disconnection and be routed to one or more alternate sites before its routes converge to a new site. We calculate two metrics, the *reconnection time* as the delay from our prefix withdrawal until we first receive a ping response from the target at any site and the *failover time* as the delay from our prefix withdrawal until the first ping response after which the target does not switch sites or experience disconnection again. Reconnection time describes how quickly a client can connect to another site for the first time after the site goes down and is the lower bound of the time to restore the service. However, following this initial reconnection time, some clients may experience further periods of disconnection or switch sites, which can break ongoing connections. To quantify the above effect, we measure the failover time, which serves as a conservative upper bound for the time to restore the service after failure, since client transactions can succeed before this time as long as they are not interrupted by a site change. A small number of clients change anycast sites frequently [44], which could inflate our tail failover times.

Figure 2 presents the CDF of reconnection and failover time for `proactive-superprefix`, `reactive-anycast`, `proactive-prepending`, and anycast. The CDF is across all targets for all sites. Our two techniques `reactive-anycast` (§4) and `proactive-prepending` (§4) have a reconnection time close to anycast, with the median around 10 seconds. `reactive-anycast` has a very similar failover time as anycast. `proactive-prepending` has a failover time ~5 seconds slower than anycast, which is likely because the longer routes are less preferred during the convergence. `proactive-superprefix` (§3) has a much longer failover time than anycast (§3) or our new techniques (§4). A recent study observed many clients initiating connections to servers long after the TTL of the corresponding DNS record expired (890s in the median) [1], suggesting that the tail failover for unicast is likely much slower than for anycast or our techniques.

For each technique, the gap between reconnection and failover time (a few seconds to 20 seconds at median) shows that clients may bounce between sites for a short period of time after they reconnect for the first time, with most targets bouncing once or twice. We also find that, during this interval, most targets do not experience periods of unreachability. Infrequent bounces and high availability mean short connections are unlikely to be interrupted unless they overlap the time when the client switches sites. A long connection may be interrupted during this interval but the connection can be re-established by applications. Since many CDN connections are short and idle for much of their duration [35], we think that restoring reachability quickly (reconnection time) while having long connections potentially being interrupted (before failover time) is a better solution than waiting for unpredictable DNS caching to be cleared.

Finally, we evaluate each technique twice using different sets of targets selected under the same criterion (§5.1) and observe similar reconnection and failover time.

5.4.2 Traffic control. The techniques `reactive-anycast` (§4) and `proactive-superprefix` (§3) can route all targets to a specific site because the prefix is unicast in normal operation. The technique `proactive-prepending` loses some traffic control (§4) in exchange for higher availability and the lower risk of not requiring global reconfiguration in the face of a failure (§7). To further investigate the tradeoff between control and the fast failover that provides high availability, we conduct two experiments: one prepends three times at other sites, and another one prepends five times at other sites. We measure the fraction of clients that are routed to the intended site and the failover time when the specific site withdraws the prefix.

Table 1 shows the fraction of targets that are routed to the specific site. Most sites can attract ~60% of the clients that cannot be routed to it by anycast (plus all those that can). Three of the sites (`bos`, `msn`, and `atl`) see obvious improvement when increasing the number of prepends. However, as the number of prepends increases, the

	ams	ath	bos	atl	sea1	slc	sea2	msn
Not routed by anycast	15%	90%	80%	95%	87%	80%	69%	80%
prepend 3	55%	97%	58%	58%	6%	57%	78%	28%
prepend 5	54%	95%	69%	75%	6%	64%	87%	68%

Table 1: For targets that are within 50 ms of a site, % that anycast routes to a different site (2nd row). Of those targets that anycast routes to different sites, % that can be routed by `proactive-prepending` when other sites prepend 3 or 5 times (lower rows). Section 5.1 presents target selection criterion.

failover time slightly increases, likely because longer alternative paths are less preferred during convergence (appendix C.2).

The degree to which increasing prepends improves control depends on specifics of the sites [28]. We investigated why many clients route to sites announcing prepended routes, especially when `sea1` was the non-prepending (*i.e.,* intended) site. We found that 82% of targets route to another site rather than `sea1` because the other route is preferred by standard BGP policy (*e.g.,* it was via a customer rather than a peer). Appendix C.1 provides details.

This lack of control will not impact CDNs that follow our recommendation to only announce the prepended route for a site's prefix to neighbors that also connect to the site and hence receive the non-prepended route (§4). Further, even without this recommendation, this scenario is unlikely to significantly impact large CDNs. First, the providers of large CDNs tend to be tier-1 providers or large tier-2/regional providers [38]. These providers are unlikely to have providers they export these routes to, and so only the direct providers of the CDNs will have customer routes. Second, if a network has any peer routes, it likely has a peer route to the non-prepended (intended) site. The only networks with peer routes to large CDNs are direct peers and peers (generally tier-1s) of large providers of the CDN. According to conversations with CDN operators, these large providers generally connect to the CDN in all regions where both have a large presence, meaning they will learn (and prefer) the non-prepended route for all sites. Similarly, a peer of the CDN generally peers with the CDN in all regions in which it has a presence, and hence is likely to receive non-prepended routes for all sites in those regions. In the future, we will investigate the traffic control of `proactive-prepending` on real CDNs.

5.4.3 Result generalization. PEERING lacks the global footprint and connectivity of real CDNs, and so we consider factors that influence whether our results generalize. For `proactive-superprefix`, the withdrawal convergence is the deciding factor in failover time, because only after that can the routes to the covering prefix be used. For `reactive-anycast`, the propagation of new valid anycast routes is the key to failover, because the new valid routes can replace the invalid routes caused by withdrawals. In Appendix A and Appendix B, we show that withdrawal convergence and announcement propagation speed of PEERING prefixes has a similar time to other networks (including hypergiants). This suggests that the failover result of `proactive-superprefix` and `reactive-anycast` can be generalized to real CDNs. For `proactive-prepending`, we explain in Section 4 which sites a real CDN should announce the prepended routes to maximize the traffic control while providing alternative routes for fast failover.

Technique	Control	Availability	Risk
`proactive-prepending`	medium	high	low
`reactive-anycast`	high	high	high
`proactive-superprefix`	high	medium	low
anycast	low	high	low
unicast	high	low	low

Table 2: CDN redirection techniques tradeoffs. Figure 2 provides quantitative comparisons of availability.

6 RELATED WORK

CDN traffic control and availability. Previous work shows how DNS and DNS extensions can map clients to a nearby site [9]. The effects of DNS caching and violations of DNS TTL have been studied [1, 21, 29]. They prevent clients from getting the up-to-date DNS records and reduce CDN availability. These effects motivate our investigation of techniques that provide failover for cached DNS records. Egress path selection has been studied to address failures on the egress path [25, 36, 45]. Our work is complementary, aiming to improve failure recovery on the ingress path through site selection. Other work uses techniques similar to our `proactive-prepending` to ensure anycast availability during DDOS attack [32]. FastRoute uses DNS to divert traffic from anycast frontends to a separate anycast deployment of datacenters for load balancing and failover [13], but within each deployment it has the limited control of anycast.

Routing events and BGP convergence. Previous work has found that withdrawals tend to take longer than announcements to converge, with a median of 170 seconds [24]. We use these insights to explain the limitations of `proactive-superprefix` (§3) and develop techniques that avoid waiting for a unicast prefix withdrawal to converge. Previous work observed that anycast failover completes within 20 seconds for many clients [3], and our techniques leverage this fast failover (§4). Other work assesses loss during route changes [43], which is related to our experiments.

7 CONCLUSION

Existing techniques to direct users to CDN sites for latency-sensitive services cannot achieve both fast failover and precise control. While unicast lacks fast failover and anycast lacks traffic control, we proposed techniques that combine strengths of both. We evaluated the techniques on the real Internet by emulating a small-scale CDN. Our techniques achieve a combination of traffic control and fast failover that existing techniques cannot match.

Table 2 compares the techniques. A technique has high client-to-site control if it has the same control as unicast, low control if it is the same as anycast, and medium control if it is between anycast and unicast. A technique's availability is high if its failover time is close to anycast's, low if it depends on new DNS record distribution, and medium if it improves the availability of unicast but is still slower than anycast. A technique has high risk if it requires global routing reconfiguration after a site failure; otherwise it has low risk. Our techniques achieve better control than anycast and/or better availability than unicast, and represent different tradeoffs.

Acknowledgements. The NSF partially funded this project via grant CNS-1835253. Italo is partially funded by FAPESP APR-20/05192-9. We thank our shepherd Cecilia Testart and the anonymous reviewers for their insightful comments.

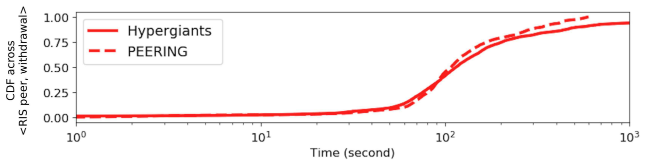


Figure 3: Convergence time for unicast prefix withdrawal from hypergiants and PEERING. They have similar convergence time at median and tail.

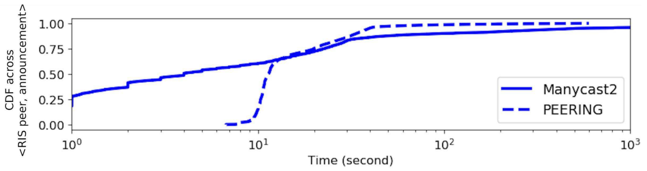


Figure 4: Propagation time of anycast announcements of Manycast^2 and PEERING prefixes. They share similar results at median and tail.

A CONVERGENCE OF UNICAST PREFIX WITHDRAWALS

Following a site failure, `proactive-superprefix` does not failover until the withdrawal of the site's unicast prefix convergences, at which point routers will use the covering superprefix (§3). To assess the likely delay, we study the convergence time of withdrawals of announcements from hypergiants, which have similar deployments and connectivity to CDNs (and in some cases are CDNs). We obtain a list IP addresses of *on-net* web servers *within* hypergiant ASes from a HTTPS scan in 2021 also used in a recent study of hypergiants *off-net* (*i.e.,* outside the ASes) deployments [14]. Each on-net IP address belonged to some hypergiant and hosted HTTPS services. We remove all anycast addresses, according to 2021 data from Manycast^2 [39]. For the remaining unicast addresses, we then use RIPE Routing History API [31] to find the contemporaneous longest IP prefixes for those addresses and the *visibility* of those prefixes, which we define as the fraction of RIS peers that have routes to the prefix (out of RIS peers that export full BGP tables). RIPE Routing History aggregates data by the day and so even a fully withdrawn prefix may not show visibility 0 if the withdrawal does not span a full day. We flag as potentially withdrawn a prefix that previously had visibility > 0.9 and then experiences a day with reduced visibility of < 0.7. We then download the BGP updates of the prefix around the potential withdrawal time (one day before to one day after) from RIS collectors and verify whether they are actual withdrawals by checking whether 90% of the peers eventually withdraw the route. We do not know the exact time the site withdrew the prefix, so we estimate it as the first time when 5 withdrawals are seen within 20 seconds. To verify this criterion, we withdraw prefixes from PEERING sites and find that the difference between the PEERING withdrawal and the estimated time is within 10 seconds at median.

For a given withdrawal, we compute the convergence time of a BGP collector peer as the time between the estimated withdrawal time and the last update from that peer (in a 1000s window after the withdrawal time). Figure 3 depicts the distribution of convergence times per ⟨`RIS peer, hypergiant withdrawal`⟩, with a median of 100s and a 90th percentile of 400 seconds. This result suggests that `proactive-superprefix` can take hundreds of seconds to failover, and this delayed convergence can be longer than the DNS TTLs used by CDNs, meaning that `proactive-superprefix` may provide limited benefit over the status quo. Figure 3 also depicts withdrawal times for unicast prefix announced and then withdrawn from PEERING, which yields a very similar distribution to the hypergiant withdrawals. This similarity suggests that the convergence/failover time results in Section 5.4.1 that use PEERING may be similar to what a real CDN would experience.

B ANYCAST ANNOUNCEMENT PROPAGATION

The deciding factor for `reactive-anycast` failover is how quickly an anycast announcement can propagate, so we measure how quickly anycast announcements propagate on the Internet to estimate how quickly `reactive-anycast` would failover if used by real networks. Rather than limiting our analysis to hypergiant prefixes, we use all anycast prefixes because we observed few anycast announcements from hypergiants, and hypergiants tend to have particularly short AS paths [2, 10], and so our use of a broader set of anycast networks will provide a conservative (over)estimate of the failover time if a CDN were to use `reactive-anycast`. We obtain anycast addresses from the Manycast^2 census result [39]. We use RIPE Routing History API [31] to find instances when a prefix becomes announced (visibility > 0.9) after a period in which it was not (visibility is zero). We download the BGP updates of the prefix during the potential announcement period from RIS collectors. We estimate the anycast announcement as having occurred when 5 announcements are made by route collector peers in 20 seconds (the burst of announcements is likely to be caused by the anycast announcement). We have assumed that the announcements were anycast when they were made (since the prefixes are from Manycast^2 anycast prefixes dataset), but we did not have a way of verifying this.

Figure 4 shows the anycast announcement distribution of announcement propagation time per ⟨`RIS peer, announcement`⟩ for the anycast prefixes identified by Manycast^2 as well as anycast announcements we make from PEERING. For both sets of announcements, the median delay is less than 10s, suggesting that `reactive-anycast` is likely to introduce alternative valid routes to networks much earlier than the invalid ones are eliminated in `proactive-superprefix` (appendix A).

C ADDITIONAL ANALYSIS ON PROACTIVE-PREPENDING

C.1 Explaining poor control

This section investigates the results of Table 1 to understand why many clients route to sites announcing prepended routes, especially when `sea1` is the non-prepending (*i.e.,* intended) site but only attracted responses from 6% of targets. To summarize, we use reverse traceroute [41] to measure the routes and find that 82% of targets route to another site rather than `sea1` because the other route is preferred according to standard BGP policy (*e.g.,* via a customer rather than a peer). For 54% of targets (including a subset of the 82%), providers prefer to route through an R&E network to another site, rather than through a commercial network to `sea1`.

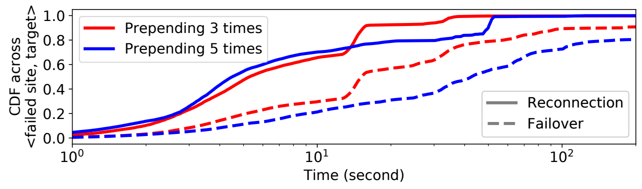


Figure 5: Prepending 3 and 5 times for `proactive-prepending`. Prepending more times in other sites can lead to longer reconnection and failover time.

C.1.1 Experiment. We announce a unicast prefix u from `sea1` and an anycast prefix a_5 from all sites including `sea1`, with all other sites prepending five times. We then run reverse traceroutes from the 50k `sea1` targets to the two prefixes. Of the 50k targets, we successfully measure 17,908 pairs of reverse paths to both u and a_5, so we can compare them and analyze the reasons why only 6,479 (36.2%) of the targets selected `sea1` for a_5. For the remaining targets, reverse traceroute failed to measure the paths because they did not support the Record Route IP option, which reverse traceroute relies on to measure the paths.

C.1.2 Methodology. We use standard IP-to-AS mapping to translate the reverse traceroutes to AS-level paths. We compare each target's route to u with its route to a_5, identify the last common AS after which paths diverge (termed the *diverging AS*), and compare the two AS links defining the divergence. We look at two things: the type of the next hop AS, using a state-of-the-art classification [46], and the type of AS relationship between the diverging AS and the divergent options it selects, using the CAIDA relationships dataset [6].

C.1.3 Results. First, no target has an AS path to u that is more than five longer than its path to a_5, meaning that AS path length is likely not the determining factor in deciding not to route to `sea1` for a_5. Of the 11,429 targets that did not go to `sea1`, for 6,169 (54%), the next hop of the diverging AS to a_5 is an R&E network, whereas the unicast AS path goes through a commercial AS. An example is when the diverging AS is Level3, and the unicast AS path goes to u via AS2914 (NTT), whereas it goes to a_5 via the Pacific Northwest Gigapop to `sea2` site (at University of Washington), rather than to `sea1` (at the Seattle Internet exchange). Of the 4,866 pairs of AS links for which we could infer relationships (the other pairs containing at least one AS link with no classification in the AS relationships dataset), 3,986 (82%) are likely explained by business preferences for customer links over peer links over provider links, with the diverging AS using a more preferred link to reach a_5 compared to the link used to reach u and `sea1`. These patterns are largely a result of PEERING's unusual properties as an academic testbed with ad hoc hosting via volunteer networks and, as we discuss in Section 5.4.2, we do not expect that real CDNs would experience them frequently and so would likely retain more control with `proactive-prepending` than we did.

C.2 `proactive-prepending` failover time and number of prepends

We conduct two experiments for `proactive-prepending`. In normal operation, we prepend at other sites 3 times in one experiment and 5 times in another experiment. The traffic control improves at a few PEERING sites when the number of prepends increases (§5.4.2). Figure 5 shows the reconnection and failover time of the two configurations. The reconnection time remains similar but the failover time increases by 20 seconds at median when increasing the number of prepends from 3 to 5. This result demonstrates that there is a tradeoff between the degree of traffic control and failover time.

REFERENCES

[1] Mark Allman. Putting DNS in Context. In *ACM IMC*, 2020.
[2] Todd Arnold, Jia He, Weifan Jiang, Matt Calder, Italo Cunha, Vasileios Giotsas, and Ethan Katz-Bassett. Cloud Provider Connectivity in the Flat Internet. In *ACM IMC*, 2020.
[3] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. A Measurement-Based Deployment Proposal for IP Anycast. In *ACM IMC*, 2006.
[4] Ilker Nadi Bozkurt, Anthony Aguirre, Balakrishnan Chandrasekaran, P Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. Why is the Internet so Slow?!. In *PAM*, 2017.
[5] Sam Burnett, Lily Chen, Douglas A. Creager, Misha Efimov, Ilya Grigorik, Ben Jones, Harsha V. Madhyastha, Pavlos Papageorge, Brian Rogan, Charles Stahl, and Julia Tuttle. Network Error Logging: Client-side Measurement of End-to-end Web Service Reliability. In *USENIX NSDI*, 2020.
[6] CAIDA. 2020. AS Relationships. https://www.caida.org/catalog/datasets/as-relationships/
[7] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. Analyzing the Performance of an Anycast CDN. In *ACM IMC*, 2015.
[8] Matt Calder, Ryan Gao, Manuel Schröder, Ryan Stewart, Jitendra Padhye, Ratul Mahajan, Ganesh Ananthanarayanan, and Ethan Katz-Bassett. Odin: Microsoft's Scalable Fault-Tolerant CDN Measurement System. In *USENIX NSDI*, 2018.
[9] Fangfei Chen, Ramesh K. Sitaraman, and Marcelo Torres. End-User Mapping: Next Generation Request Routing for Content Delivery. In *ACM SIGCOMM*, 2015.
[10] Yi-Ching Chiu, Brandon Schlinker, Abhishek Balaji Radhakrishnan, Ethan Katz-Bassett, and Ramesh Govindan. Are We One Hop Away from a Better Internet?. In *ACM IMC*, 2015.
[11] Xun Fan and John Heidemann. Selecting Representative IP Addresses for Internet Topology Studies. In *ACM IMC*, 2010.
[12] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. Guidelines for Interdomain Traffic Engineering. In *ACM SIGCOMM CCR*, 2003.
[13] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. Fastroute: A scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *USENIX NSDI*, 2015.
[14] Petros Gigis, Matt Calder, Lefteris Manassakis, George Nomikos, Vasileios Kotronis, Xenofontas Dimitropoulos, Ethan Katz-Bassett, and Georgios Smaragdakis. Seven Years in the Life of Hypergiants' off-Nets. In *ACM SIGCOMM*, 2021.
[15] Vasileios Giotsas, Christoph Dietzel, Georgios Smaragdakis, Anja Feldmann, Arthur Berger, and Emile Aben. Detecting Peering Infrastructure Outages in the Wild. In *ACM SIGCOMM*, 2017.
[16] Vasileios Giotsas, Matthew Luckie, Bradley Huffaker, and kc claffy. Inferring Complex AS Relationships. In *ACM IMC*, 2014.
[17] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Evolve or die: High-availability Design Principles Drawn from Googles Network Infrastructure. In *ACM SIGCOMM*, 2016.
[18] Hang Guo and John Heidemann. Detecting ICMP Rate Limiting in the Internet. In *PAM*, 2018.
[19] Rebecca Hersher. 2017. Amazon and the $150 Million Typo. https://www.npr.org/sections/thetwo-way/2017/03/03/518322734/amazon-and-the-150-million-typo
[20] Weifan Jiang, Tao Luo, Thomas Koch, Yunfan Zhang, Ethan Katz-Bassett, and Matt Calder. Towards Identifying Networks with Internet Clients Using Public Data. In *ACM IMC*, 2021.
[21] Jaeyeon Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS Performance and the Effectiveness of Caching. In *ACM SIGCOMM IMW*, 2001.
[22] ISI ANT Lab. 2022. IPv4 Hitlists. https://ant.isi.edu/datasets/ip_hitlists/
[23] ISI ANT Lab. 2022. Verfploeter: Active Measurement of Anycast Catchements. https://ant.isi.edu/software/verfploeter/index.html
[24] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed Internet Routing Convergence. In *ACM SIGCOMM*, 2000.
[25] Raul Landa, Lorenzo Saino, Lennert Buytenhek, and João Taveira Araújo. Staying Alive: Connection Path Reselection at the Edge. In *USENIX NSDI*, 2021.
[26] Zhihao Li. 2019. *Diagnosing and Improving the Performance of Internet Anycast.* Ph.D. Dissertation. University of Maryland, College Park.
[27] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Internet Anycast: Performance, Problems, & Potential. In *ACM SIGCOMM*, 2018.

[28] Pedro Marcos, Lars Prehn, Lucas Leal, Alberto Dainotti, Anja Feldmann, and Marinho Barcellos. AS-Path Prepending: There is No Rose without a Thorn. In *ACM IMC*, 2020.

[29] Giovane C. M. Moura, John Heidemann, Ricardo de O. Schmidt, and Wes Hardaker. Cache Me If You Can: Effects of DNS Time-to-Live. In *ACM IMC*, 2019.

[30] RIPE NCC. 2022. Routing Information Service (RIS). https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris

[31] RIPEstat. 2021. Routing History. https://stat.ripe.net/docs/02.data-api/routing-history.html

[32] A S M Rizvi, Leandro Bertholdo, João Ceron, and John Heidemann. Anycast Agility: Network Playbooks to Fight DDoS. In *USENIX Security*, 2022.

[33] Mark Russinovich. 2020. Advancing safe deployment practices. https://azure.microsoft.com/en-us/blog/advancing-safe-deployment-practices/

[34] Brandon Schlinker, Todd Arnold, Italo Cunha, and Ethan Katz-Bassett. PEERING: Virtualizing BGP at the Edge for Research. In *ACM CoNEXT*, 2019.

[35] Brandon Schlinker, Italo Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. Internet Performance from Facebook's Edge. In *ACM IMC*, 2019.

[36] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V. Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. In *ACM SIGCOMM*, 2017.

[37] Kyle Schomp, Onkar Bhardwaj, Eymen Kurdoglu, Mashooq Muhaimen, and Ramesh K Sitaraman. Akamai DNS: Providing Authoritative Answers to the World's Queries. In *ACM SIGCOMM*, 2020.

[38] Rachee Singh, Sharad Agarwal, Matt Calder, and Paramvir Bahl. Cost-effective Cloud Edge Traffic Engineering with Cascara. In *USENIX NSDI*, 2021.

[39] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, KC Claffy, and Anna Sperotto. MAnycast2: Using Anycast to Measure Anycast. In *ACM IMC*, 2020.

[40] Srikanth Sundaresan, Nazanin Magharei, Nick Feamster, Renata Teixeira, and Sam Crawford. Web Performance Bottlenecks in Broadband Access Networks. In *ACM SIGMETRICS*, 2013.

[41] Kevin Vermeulen, Ege Gurmericliler, Italo Cunha, Dave Choffnes, and Ethan Katz-Bassett. Internet Scale Reverse Traceroute. In *ACM IMC*, 2022.

[42] Route Views. 2022. University of Oregon Route Views Project. http://www.routeviews.org/routeviews/

[43] Feng Wang, Zhuoqing Morley Mao, Jia Wang, Lixin Gao, and Randy Bush. A Measurement Study on the Impact of Routing Events on End-to-End Internet Path Performance. In *ACM SIGCOMM*, 2006.

[44] Lan Wei and John Heidemann. Does Anycast Hang Up on You? In *Network Traffic Measurement and Analysis Conference (TMA)*, 2017.

[45] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *ACM SIGCOMM*, 2017.

[46] Maya Ziv, Liz Izhikevich, Kimberly Ruth, Katherine Izhikevich, and Zakir Durumeric. ASdb: A System for Classifying Owners of Autonomous Systems. In *ACM IMC*, 2021.