# 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 access to clients by replicating content at geographically distributed sites. Most CDNs route clients to a site using anycast or unicast with DNS-based redirection. Anycast compromises control of user-to-site mapping (and hence performance), and unicast compromises availability, two fundamental goals of CDNs. We first analyze anycast and unicast and explain why neither of them satisfies both goals. We then present new hybrid techniques and demonstrate through experiments on the real Internet that we can support a high level of traffic control while greatly improving availability following site failures.

**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 with a large set of 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 [15], hardware failures (e.g., router, line-card), network misconfiguration (e.g., BGP, DNS), facility outages [13], or sudden bursts of traffic (e.g., DDoS). 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.

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 trade-off between traffic control and availability. With IP unicast, a CDN directs users to a specific site by using DNS to return IP addresses from the site, but the speed at which the CDN can move users between sites is inherently limited by DNS record time-to-live (TTL). Setting TTLs too low introduces additional latency for applications, and some client software and DNS resolvers continue to use a DNS record after the TTL expires, continuing to direct traffic to the corresponding site [1, 19, 26]. IP anycast relies on BGP's distributed path selection to direct clients to sites and so lacks unicast's control [24, 25] but enables fast failover in the presence of failures simply by withdrawing announcements of the anycast address from the failed site [2].

To address the current need to give up 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 approach relies on each site being assigned a distinct prefix (*e.g.,*/24 or /48), from which DNS records are returned as in (Nits: A-Comment-P10) unicast. To achieve fast site failover, in addition to changing 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 t 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, the packets destined to the failing site can still be routed to another site.

We evaluate our techniques with Peering [31], a testbed that lets us make BGP announcements and exchange traffic with the real Internet at different sites. Our results 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 have 10 seconds median failover time (only 2 seconds longer than IP anycast). In contrast, the DNS TTLs used by top domains are around 10 minutes at median [26]. Although major CDNs use much lower DNS TTLs such as tens of seconds, especially those that use unicast, (Minor issue: Top-domain TTL may not be representative) DNS records are cached on resolvers and clients, and applications may violate TTL and use outdated records [1].
- Our second technique (§4) can redirect 60% of the clients to the sites that IP anycast cannot route to (Table 1), while achieving roughly the same failover time as IP anycast.

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

## 2 BACKGROUND

In this section, we review the two primary techniques for directing clients to sites for latency-sensitive services.

*IMC '22, October 25–27, 2022, Nice, France*
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9259-4/22/10…$15.00
https://doi.org/10.1145/3517745.3561421

**Unicast.** In DNS-based redirection [8], 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 have to use DNS to point users to sites.

During failures, the CDN updates DNS records and relies on the clients requerying DNS for IP addresses to other sites, but each record is cached by the client's recursive resolver, OS, and potentially application. 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 be a small value (*e.g.,* less than 60s), this impacts latency in applications [3, 36], and it can be violated. Recent measurements found that 22.2% of connections were established after the TTL expired, and the median time since expiration was 890s [1]. Another study on DNS caching showed that decreasing TTL to very small values does not reduce cache hit rate [19]. This means that many clients may not get redirected quickly during a site failure even if the CDN has already updated its DNS records.

**IP Anycast.** Anycast networks take advantage of a simpler operational model than DNS, where 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 although anycast CDN operates a large set of sites and peering connections, a subset of the clients are routed to suboptimal sites [6, 25]. Despite this, IP anycast has some advantages during site failure: no DNS records need updating, and even if one route is withdrawn by a particular site due to a failure, clients are rerouted to other sites quickly. Although the failover time is subject to BGP convergence, previous work [2] 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.

**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 [6]. 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.

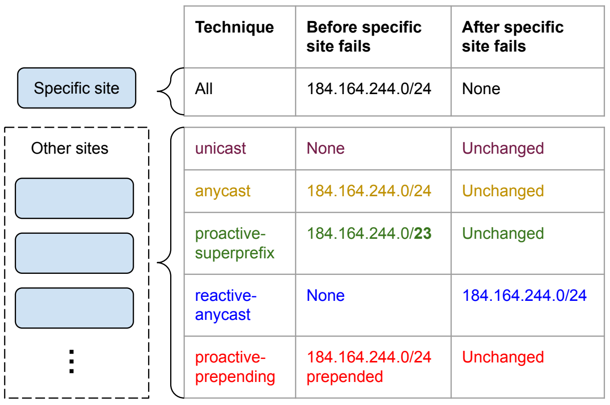
Second, a CDN can announce a unicast prefix from a particular site while also announcing a less-specific covering prefix from other sites. For example, a specific site advertises 184.164.244.0/24, and all sites advertise 184.164.244.0/23 as backup (`proactive-superprefix` in Figure 1). (Major issue: Better illustration of experiment setup for each technique) 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 [27] BGP dump for prefixes containing IP addresses of hypergiant web servers from April 2021 [12] (Nits:B-Comment-P4). Of the most specific prefixes that hosted these servers and were announced by the hypergiants, 39% (12% to 95% per hypergiant) were also covered by less specific prefixes announced at the same time.

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 [22] due to the lack of valid alternative routes to 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. As BGP searches exhaustively for alternate routes that do not exist, it takes 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 prefix, delaying failover. In contrast, the withdrawal of an anycast prefix from a site converges faster because alternative valid paths to other sites are already learned by networks and will quickly terminate BGP's search for alternate routes (§5.4.1). (Nits: B-Comment-P6)

We study the convergence time for unicast prefixes on the Internet and find that it takes ~100s to converge at the median and more than 10 minutes at the tail (appendix A). Given that the median DNS TTL for popular domains is 10 minutes [26] 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 [15] of a CDN that hosts important/popular Internet services.

## 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 [6]. 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



| 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.**

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 [6].

**`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 this approach is similar to anycast: both rely on other networks replacing the invalid paths with valid paths to other sites, but is different than anycast as follows: in anycast the alternative valid paths are already learned by networks before failure, but 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) 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). During convergence, 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.

This technique 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, but CDNs have deployed such systems to quickly detect problems [4, 7]. 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 [11, 33]. (Major issue: CDN real-time monitoring and action) However, a disadvantage of this approach is that global routing configuration must be updated in response to a failure. Such simultaneous global configuration changes are operationally treacherous [30], potentially resulting in unexpected and cascading routing changes and introducing the potential for a global outage from a simple mistake [17]. To debug the propagation of the new anycast announcement, prior to failure, a CDN can rotate through its sites, announce a test prefix at all sites but not the current one to see if its clients are routed as expected. (Major issue: CDN real-time monitoring and action)

**`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.

This technique 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 more invalid routes following a failure, so it may take longer for them to be preferred, delaying failover. We will present results about traffic control and failover 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 [10], 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. It is also known that networks tend to set the same policy on peering points in the same continent [14]. (Major issue: Applying proactive-prepending to real CDNs) 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, and it peers with them at tens of locations [34]. (Major issue: Applying proactive-prepending to real CDNs) 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 [8]. (Major issue: Applying proactive-prepending to real CDNs) In addition, with modern intent-based centralized configuration management, managing configurations that are site- and/or peer-specific is straightforward [31], so adding or removing a peer introduces only manageable and automatable configuration changes. (Major issue: Applying proactive-prepending to real CDNs)

## 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 [31] 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. (Minor issue: Comparison to DNS-based) 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, (Nits: C-Comment-P6 ) 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 during normal operation and availability during failure, we select different sets of client targets for each Peering site from ISI's IPv4 Hitlist [20]. The Hitlist provides one IPv4 address per /24 that is likely to respond to pings, and there are ~3.5M actually responsive addresses after our verification. Probing targets from Peering provides better network coverage and more frequent measurements than possible from platforms such as RIPE Atlas. From the responsive targets, we choose 2.8M in prefixes that have web clients [18], as these 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 is enough 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. We claim that if a target is routed to the site by anycast, it can be routed to the site by `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 the 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 the criterion.

Using the criteria above, we select 50,000 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 20,000 ASes. (Major Issue: Target Selection)

### 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. 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, record which Peering site they are routed to, and retain the responsive targets that are routed to the current site only, which represent the targets the technique can control. 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

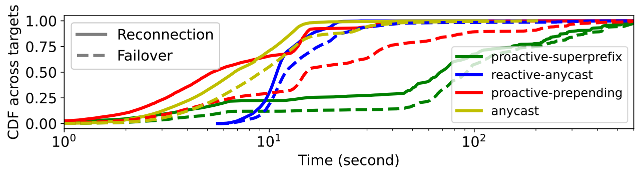


**Figure 2: Reconnection and failover time for each technique. `reactive-anycast` and `proactive-prepending` achieve failover time similar to anycast. `proactive-superprefix` has a much longer failover time than anycast.**

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 send the ping requests using Verfploeter [21] 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). (Major issue: Failover time definition and calculation) Finally, we collect BGP feeds from RIS [27] and Routeviews [38] 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 is ethical for the following reasons. First, these targets have not opted out during ISI's scan [9]. 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, our probing rate is within the ICMP rate limiting of most /24 blocks [16]. Last, deployments of our techniques on CDNs (§4) do not require constant and frequent probing. The probing is needed for the evaluation of reconnection and failover time in our study only. (Major Issue: Ethics)

## 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 effect of switching sites after failure, 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.

Figure 2 presents the CDF of reconnection and failover time for `proactive-superprefix`, `reactive-anycast`, `proactive-prepending`, and anycast. For `proactive-prepending`, we prepend three times at other sites here (Section 5.4.2 compares results for three vs. five, we pick 3 here since majority of the sites do not see much change in traffic control). The CDF is across all targets for all sites. Our two techniques `reactive-anycast` (§4) and `proactive-prepending` (§4) are able to reach a reconnection time close to anycast, with the median around 10 seconds. `reactive-anycast` has the same failover time as anycast and `proactive-prepending` has a failover time ~5 seconds slower than anycast. `proactive-prepending` has a longer failover time than anycast at the tail, and we think that this is 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. Based on the impact of DNS TTL violation [1], in unicast, more than 20% of the clients would have median failover time as long as 890s. All techniques we measured achieve a shorter failover time than unicast at tail. (Minor issue: Comparison to DNS-based)

For each technique, the difference between reconnection and failover time (a few seconds to 20 seconds at median) shows that before the final failover, 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. During destination site change, we also find that most targets are able to reach one of the sites. Infrequent bounces and high availability mean short connections are unlikely to be interrupted unless they overlap the time when the destination site changes. A long connection may be interrupted between reconnection and failover but the connection can be re-established by applications. Since many CDN connections are short and idle for much of their duration [32], 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. (Minor issue: Consistency of result)

*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). (Major issue: More investigation on pros/cons/comparisons of techniques) 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 prefix is withdrawn from the specific site.

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 to the site, how many percentages of them are routed by anycast to other sites (top row). Of those targets that anycast routes to other sites, how many percentages of them 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 (see Appendix C.2 for full details).

We investigated why many clients route to sites announcing prepended routes, especially when `sea1` was the non-prepending (*i.e.,* intended) site but only received 6% of targets. We found that 82% of targets route to another site rather than `sea1` because the other route is preferred according to 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 (and should prefer) the non-prepended route (§4). Further, even without this recommendation, this scenario is unlikely to significantly impact large CDNs. First, the only providers of large CDNs tend to be tier-1 providers or, in certain regions, large tier-2/regional providers [34]. (Nits: B-Comment-P10) 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, networks are unlikely to have a prepended peer route but only a provider route to the non-prepended site that a CDN wants to serve the network from. The only networks with peer routes to large CDNs are direct peers and peers (generally tier-1s) of large providers of the CDN. But, according to conversations with CDN operators, these large providers will generally connect to the CDN in all regions where they both have a large presence, meaning they will learn (and prefer, and propagate) the non-prepended route for all sites in the region. Similarly, a peer of the CDN will generally peer 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 the regions where the CDN is likely to want to direct it. In the future, we will investigate how much traffic control `proactive-prepending` can give CDNs under different announcement scenarios.

*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.

## 6 RELATED WORK

**Egress Path Failover.** CPR provides connection granularity path failover by monitoring connections and migrating those that stall to alternate egress paths from the same site [23]. General egress traffic engineering solutions such as Espresso [40] and EdgeFabric [33] can also route around failures by using a different egress path at a site but operate at a prefix granularity. Egress path selection can only address failures that occur in one direction so our work is complementary in that we aim to improve failure recovery on the ingress path through site selection.

**Routing events and BGP convergence.** Previous work has measured the convergence time of different routing events and found that withdrawals tend to take the longest time to converge, with a median of 170 seconds [22]. We use these insights to explain the limitations of `proactive-superprefix` (§3) and develop techniques that avoid waiting on for a unicast prefix withdrawal to converge. Previous work observed that anycast failover completes within 20 seconds for many clients [2], and we leverage this property of anycast in our techniques (§4). Other work shows the burst of losses during routing events [39], which is related to our experiments.

**CDN traffic control and availability.** Much previous work has been done to control CDN traffic for better performance. Previous work shows how unicast can map clients to a proximal site to improve performance on clients [8]. The effect of DNS caching has been studied [19, 26], including work that reported on violations of DNS TTL [1]. These effects motivate our investigation of techniques that provide failover for cached DNS records. Other work presents techniques to ensure anycast CDN availability during DDOS attack [29]. FastRoute is a system that enables some load control with anycast by offloading traffic from edge sites to high capacity data centers [11]. FastRoute's main disadvantage is that it sacrifices backbone capacity to route CDN traffic to datacenters. FastRoute operators explained to us that this tradeoff of increasingly valuable backbone capacity was seldom worthwhile.

## 7 CONCLUSION

This paper studied approaches for CDNs to achieve fast failover without giving up traffic control. We described why existing techniques are insufficient: unicast lacks fast failover, and anycast lacks traffic control. We proposed techniques that leverage the strengths of both unicast and anycast. We evaluated the techniques on the real Internet by emulating a small-scale CDN and found that our techniques are able to achieve a combination of traffic control and fast failover that existing techniques cannot achieve.

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 does not have more than anycast, and medium control if it is between anycast and unicast. A technique has high availability if its failover time is close to anycast's, low availability if it depends

| 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 trade-offs among control, availability and risk. Refer to Figure 2 for quantitative details of comparisons of availability.**

on new DNS record distribution, and medium availability 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 one site is down, as such changes run the risk of turning a single site failure into a widespread outage; otherwise it has low risk. Our techniques achieve better control than anycast and/or better availability than unicast, and represent different tradeoffs. In addition, our techniques also introduce different deployment complexity. `reactive-anycast` requires a monitoring system to detect site outage and coordinate other global sites to make announcements as soon as possible. `proactive-prepending` requires managing different prepending lengths at different sites in normal operation. If a CDN prioritizes both control and availability and has the capability to develop a robust system to monitor failures and make routing configuration changes during failure, it may prefer `reactive-anycast`. `proactive-prepending` is preferable for CDNs that need high availability and low operational risk but can afford some loss of traffic control. (Major issue: More investigation on pros/cons/comparisons of techniques)

**Acknowledgements.** This paper has been partially funded by **TBD: X**. We would like to thank our shepherd Cecilia Testart and the anonymous reviewers for their insightful comments and CDN operators for providing information about CDN operation.

## REFERENCES

[1] Mark Allman. 2020. Putting DNS in Context. In *ACM IMC*.
[2] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. 2006. A Measurement-Based Deployment Proposal for IP Anycast. In *ACM IMC*.
[3] Ilker Nadi Bozkurt, Anthony Aguirre, Balakrishnan Chandrasekaran, P Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. 2017. Why is the internet so slow?!. In *PAM*.
[4] 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. 2020. Network Error Logging: Client-side measurement of end-to-end web service reliability. In *USENIX NSDI*.
[5] CAIDA. 2020. AS Relationships. https://www.caida.org/catalog/datasets/as-relationships/
[6] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. 2015. Analyzing the Performance of an Anycast CDN. In *ACM IMC*.
[7] Matt Calder, Ryan Gao, Manuel Schröder, Ryan Stewart, Jitendra Padhye, Ratul Mahajan, Ganesh Ananthanarayanan, and Ethan Katz-Bassett. 2018. Odin: Microsoft's Scalable Fault-Tolerant CDN Measurement System. In *USENIX NSDI*.
[8] Fangfei Chen, Ramesh K. Sitaraman, and Marcelo Torres. 2015. End-User Mapping: Next Generation Request Routing for Content Delivery. In *ACM SIGCOMM CCR*.
[9] Xun Fan and John Heidemann. 2010. Selecting Representative IP Addresses for Internet Topology Studies. In *ACM IMC*.
[10] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. 2003. Guidelines for Interdomain Traffic Engineering. In *ACM SIGCOMM CCR*.
[11] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. 2015. Fastroute: A scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *USENIX NSDI*.
[12] Petros Gigis, Matt Calder, Lefteris Manassakis, George Nomikos, Vasileios Kotronis, Xenofontas Dimitropoulos, Ethan Katz-Bassett, and Georgios Smaragdakis. 2021. Seven Years in the Life of Hypergiants' off-Nets. In *ACM SIGCOMM*.
[13] Vasileios Giotsas, Christoph Dietzel, Georgios Smaragdakis, Anja Feldmann, Arthur Berger, and Emile Aben. 2017. Detecting peering infrastructure outages in the wild. In *ACM SIGCOMM*.
[14] Vasileios Giotsas, Matthew Luckie, Bradley Huffaker, and kc claffy. 2014. Inferring Complex AS Relationships. In *ACM IMC*.
[15] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2016. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *ACM SIGCOMM*.
[16] Hang Guo and John Heidemann. 2018. Detecting ICMP rate limiting in the Internet. In *PAM*.
[17] 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
[18] Weifan Jiang, Tao Luo, Thomas Koch, Yunfan Zhang, Ethan Katz-Bassett, and Matt Calder. 2021. Towards Identifying Networks with Internet Clients Using Public Data. In *ACM IMC*.
[19] Jaeyeon Jung, E. Sit, H. Balakrishnan, and R. Morris. 2001. DNS performance and the effectiveness of caching. In *ACM SIGCOMM IMW*.
[20] ISI ANT Lab. 2022. IPv4 Hitlists. https://ant.isi.edu/datasets/ip_hitlists/
[21] ISI ANT Lab. 2022. Verfploeter: Active Measurement of Anycast Catchements. https://ant.isi.edu/software/verfploeter/index.html
[22] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. 2000. Delayed Internet Routing Convergence. In *ACM SIGCOMM CCR*.
[23] Raul Landa, Lorenzo Saino, Lennert Buytenhek, and João Taveira Araújo. 2021. Staying alive: Connection path reselection at the edge. In *USENIX NSDI*.
[24] Zhihao Li. 2019. *Diagnosing and Improving the Performance of Internet Anycast.* Ph.D. Dissertation. University of Maryland, College Park.
[25] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2018. Internet Anycast: Performance, Problems, & Potential. In *ACM SIGCOMM*.
[26] Giovane C. M. Moura, John Heidemann, Ricardo de O. Schmidt, and Wes Hardaker. 2019. Cache Me If You Can: Effects of DNS Time-to-Live. In *ACM IMC*.
[27] RIPE NCC. 2022. Routing Information Service (RIS). https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris
[28] RIPEstat. 2021. Routing History. https://stat.ripe.net/docs/02.data-api/routing-history.html
[29] A S M Rizvi, Leandro Bertholdo, João Ceron, and John Heidemann. 2022. Anycast Agility: Network Playbooks to Fight DDoS. In *USENIX Security*.
[30] Mark Russinovich. 2020. Advancing safe deployment practices. https://azure.microsoft.com/en-us/blog/advancing-safe-deployment-practices/
[31] Brandon Schlinker, Todd Arnold, Italo Cunha, and Ethan Katz-Bassett. 2019. PEERING: Virtualizing BGP at the Edge for Research. In *ACM CoNEXT*.
[32] Brandon Schlinker, Italo Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. 2019. Internet Performance from Facebook's Edge. In *ACM IMC*.
[33] 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 *ACM SIGCOMM*.
[34] Rachee Singh, Sharad Agarwal, Matt Calder, and Paramvir Bahl. 2021. Cost-effective Cloud Edge Traffic Engineering with Cascara. In *USENIX NSDI*.
[35] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, KC Claffy, and Anna Sperotto. 2020. MAnycast2: Using Anycast to Measure Anycast. In *ACM IMC*.
[36] Srikanth Sundaresan, Nazanin Magharei, Nick Feamster, Renata Teixeira, and Sam Crawford. 2013. Web Performance Bottlenecks in Broadband Access Networks. In *ACM SIGMETRICS*.
[37] Kevin Vermeulen, Ege Gurmericliler, Italo Cunha, Dave Choffnes, and Ethan Katz-Bassett. 2022. Internet Scale Reverse Traceroute. In *ACM IMC*.
[38] Route Views. 2022. University of Oregon Route Views Project. http://www.routeviews.org/routeviews/
[39] Feng Wang, Zhuoqing Morley Mao, Jia Wang, Lixin Gao, and Randy Bush. 2006. A Measurement Study on the Impact of Routing Events on End-to-End Internet Path Performance. In *ACM SIGCOMM CCR*.
[40] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. 2017. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *ACM SIGCOMM*.
[41] Maya Ziv, Liz Izhikevich, Kimberly Ruth, Katherine Izhikevich, and Zakir Durumeric. 2021. ASdb: A System for Classifying Owners of Autonomous Systems. In *ACM IMC*.

## A CONVERGENCE OF UNICAST PREFIX WITHDRAWAL

Here we first give a study of the convergence time of a unicast prefix withdrawal from hypergiants. The reason we look at convergence time of a unicast prefix withdrawal is that it is directly related to the failover time of `proactive-superprefix` (§3). The convergence of

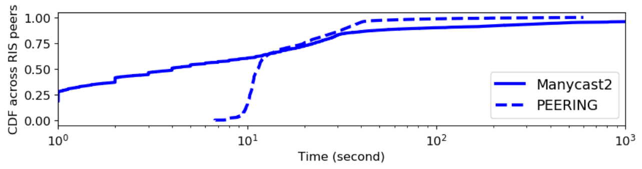


**Figure 4: Anycast announcement propagation time.**

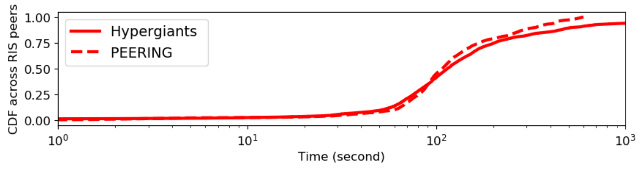


**Figure 3: Convergence time for unicast prefix withdrawal.**

a withdrawal means the route to prefix disappears, and only after that the routes to covering prefix will be used due to longest prefix matching. We look at hypergiants' prefixes because we think their deployment and connectivity are more representative of real CDNs.

We first obtain hypergiants' on-net IP addresses from a HTTPS scan in 2021 used in a recent study of hypergiants off-net deployment [12]. Each IP address belonged to some hypergiant and hosted HTTPS services. We then use RIPE Routing History API [28] to find the visibility (the fraction of RIS peers exporting full tables that have route to the prefix) of the longest prefixes in 2021 that contain those IP addresses. We then discard the anycast ones reported by Manycast2 [35] in 2021, and identify globally visible prefixes that drop their visibility or disappear. These prefix visibility drops are treated as potential withdrawals and the days they happen are considered as the potential time of withdrawals. We then download the BGP updates of the prefix around the potential withdrawal time (one day before to one day after) from RIS and Routeviews collectors, and verify that they are actual withdrawals by checking whether the majority of the peers eventually withdraw the route. We do not know the exact time of the prefix withdrawal, but estimate it as the time when 5 withdrawals are seen within 20 seconds (a burst of withdrawals is likely caused by the withdrawal from origin). We come up with this criteria by manually looking into BGP feeds during known withdrawals in experiments. To verify this criteria, we withdraw prefixes from PEERING sites. The true withdrawal time is known, but we also estimate withdrawal time using the above criteria. We find that their difference is within 10 seconds at median.

For a given withdrawal, the convergence time in each AS is computed as the timestamp of the last update in a day since the estimated withdrawal time minus the estimated withdrawal timestamp. Note that we check that most peers eventually withdraw the prefix to confirm that it is a withdrawal, we expect that it is rare for a hypergiant to repeatedly withdraw and announce the same prefix in the same day. In Figure 3 the convergence time of hypergiant prefix withdrawals on the peers of collectors (blue line) has a median of 100 seconds. The tail at 90 percentile can be as long as 400 seconds. This suggests that `proactive-superprefix` will need around 100s to failover and this amount of convergence time can be longer than many of the TTLs used by CDNs. Figure 3 also compares unicast prefix withdrawal from hypergiants and PEERING testbed and finds that they are close. This means that our results in Section 5.4.1 evaluated by PEERING can be generalized to real CDNs.

## B ANYCAST ANNOUNCEMENT PROPAGATION

The deciding factor for `reactive-anycast` failover is how quickly an anycast announcement can propagate, so we first measure the anycast announcement propagation speed on the Internet. We use the anycast announcement propagation speed to estimate whether `reactive-anycast` would failover fast if applied on real networks. We use all anycast prefixes instead of hypergiants' prefixes because (1) we do not find enough anycast announcements from hypergiants. (2) Hypergiants tend to have shorter AS paths to networks than non-hypergiant networks, and closeness to other networks would make the propagation of their prefixes faster. This means that we would have a conservative estimation on the failover time of `reactive-anycast`. We obtain anycast addresses from the Manycast2 [35] census result. We use RIPE Routing History API [28] to find a period of time when a globally invisible prefix becomes visible. We download the BGP updates of the prefix during the potential announcement period from RIS and Routeviews collectors. The timestamp of anycast announcement is estimated as the timestamp when 5 announcements are made by peers in 20 seconds (the burst of announcements is likely to be caused by the anycast announcement). We verify that it is actually an announcement by checking if peers announce the prefix before the estimated announcement timestamp.

Figure 4 shows the anycast announcement propagation time for other networks that announce Manycast2 prefixes and PEERING testbed. We observe a less than 10s delay at the median for both PEERING and Manycast2 prefixes. The fast propagation of announcements suggests 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 received 6% of targets. To summarize, we use reverse traceroute [37] 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.,* it was 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`.

*C.1.1 Experiment.* We announce a unicast prefix $u$ from `sea1`, named $u$ (for unicast) and an anycast prefix $a_5$ from all sites including `sea1`, from all other sites prepending 5 times.
We then run reverse traceroutes from the 50k `sea1` targets to the two prefixes.

On 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

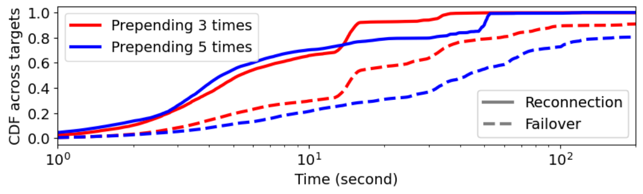


**Figure 5: Prepending 3 and 5 times for `proactive-prepending`. Prepending more times in other sites can lead to longer reconnection and failover time.**

the reasons why only 6,479 (36.2%) of the targets selected `sea1`. 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 ([41]), and the type of AS relationship between the diverging AS and the divergent options it selects, using the CAIDA relationships dataset ([5])

*C.1.3 Results.* First, the unicast AS paths is never longer than the anycast AS path plus 5 (the length of the prepending), meaning that AS path length is likely not the determining factor in deciding not to route to `sea1` for $a_5$.

On 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 AS101 (WASH-NSF-AS), therefore going 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 our 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`.

## 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. We have found that 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. We observe that the reconnection time remains the same but the failover time increases by 20 seconds at median when increasing the number of prepends from 3 to 5. This suggests that there is a tradeoff between the degree of traffic control and failover time.