Engineering Egress with Edge Fabric

Steering Oceans of Content to the World

Brandon Schlinker,[★†] Hyojeong Kim,[★] Timothy Cui,[★] Ethan Katz-Bassett,[†‡] Harsha V. Madhyastha[§]
Italo Cunha,[♯] James Quinn,[★] Saif Hasan,[★] Petr Lapukhov,[★] Hongyi Zeng[★]

[★] Facebook [†] University of Southern California [‡] Columbia University
[§] University of Michigan [♯] Universidade Federal de Minas Gerais

ABSTRACT

Large content providers build points of presence around the world, each connected to tens or hundreds of networks. Ideally, this connectivity lets providers better serve users, but providers cannot obtain enough capacity on some preferred peering paths to handle peak traffic demands. These capacity constraints, coupled with volatile traffic and performance and the limitations of the 20 year old BGP protocol, make it difficult to best use this connectivity.

We present Edge Fabric, an SDN-based system we built and deployed to tackle these challenges for Facebook, which serves over two billion users from dozens of points of presence on six continents. We provide the first public details on the connectivity of a provider of this scale, including opportunities and challenges. We describe how Edge Fabric operates in near real-time to avoid congesting links at the edge of Facebook's network. Our evaluation on production traffic worldwide demonstrates that Edge Fabric efficiently uses interconnections without congesting them and degrading performance. We also present real-time performance measurements of available routes and investigate incorporating them into routing decisions. We relate challenges, solutions, and lessons from four years of operating and evolving Edge Fabric.

CCS CONCEPTS

• **Networks** → **Traffic engineering algorithms**; *Network resources allocation*; *Network control algorithms*; *Network performance evaluation*;

KEYWORDS

Internet Routing, Border Gateway Protocol, Traffic Engineering, Software Defined Networking, Content Distribution Network

ACM Reference format:
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. In *Proceedings of ACM SIGCOMM, Los Angeles, CA, USA, August 21–25, 2017,* 14 pages.
https://doi.org/10.1145/3098822.3098853

ACM SIGCOMM, August 21–25, 2017, Los Angeles, CA, USA

ACM ISBN 978-1-4503-4653-5/17/08…$15.00
https://doi.org/10.1145/3098822.3098853

1 INTRODUCTION

Internet traffic has very different characteristics than it did a decade ago. The traffic is increasingly sourced from a small number of large content providers, cloud providers, and content delivery networks. Today, ten Autonomous Systems (ASes) alone contribute 70% of the traffic [20], whereas in 2007 it took thousands of ASes to add up to this share [15]. This consolidation of content largely stems from the rise of streaming video, which now constitutes the majority of traffic in North America [23]. This video traffic requires both high throughput *and* has soft real-time latency demands, where the quality of delivery can impact user experience [6].

To deliver this high-volume, demanding traffic and improve user-perceived performance and availability, these providers have reshaped the Internet's topology. They serve their users from numerous Points of Presence (PoPs) spread across the globe [3], where they interconnect with multiple other ASes [15]. A PoP generally has multiple paths available to reach a user network, increasing the likelihood that it has a "short" path [5]. This rich interconnectivity gives providers control over larger portions of the paths [15] and in aggregate provides necessary capacity. In contrast, the traditional Internet provider hierarchy [15] can struggle to provide the capacity needed to deliver the rapidly growing demand for content.

Although such rich connectivity potentially offers performance benefits, application providers face challenges in realizing these benefits. Strikingly, despite these massive changes in the traffic being delivered and the topology it is delivered over, the protocol used to route the traffic over the topology—the Border Gateway Protocol (BGP)—is essentially unchanged, and significant barriers exist to replacing it [5, 22]. While it is impressive that BGP has accommodated these changes, it is ill-suited to the task of delivering large volumes of consolidated traffic on the flattened topology:

- **BGP is not capacity-aware.** Interconnections and paths have limited capacity. Our measurements of a large content provider network show that it is unable to obtain enough capacity at many interconnections to route all traffic along the paths preferred by its BGP policy. A provider's routing decisions should account for these constraints, especially since a provider can *cause* congestion with huge volumes of adaptive-bitrate video traffic that expands to increase bitrate when capacity allows.
- **BGP is not performance-aware.** At any PoP, forwarding the traffic to an IP prefix along the best path chosen by BGP can lead to sub-optimal performance. The attributes that BGP relies upon for path selection, such as AS path length and multi-exit discriminators (MEDs), do not always correlate with performance.

Overriding BGP's path selection with performance-aware selection is challenging for multiple reasons. First, BGP does not incorporate performance information, and performance-aware decisions require an external mechanism to capture performance. Second, the relative performance of paths to a destination can vary over time (e.g., in response to load), but BGP changes paths only in reaction to changes in policy or the set of available routes. Third, a provider may need to measure multiple non-equivalent paths in near real-time in order to track relative performance, yet BGP traditionally permits using only a single path to reach a destination at any point in time. Fourth, because BGP uses a single path per destination, it does not natively support assigning performance-sensitive traffic and elastic traffic to different paths in order to best make use of limited capacity on the best routes.

Although these limitations are well-known, our paper discusses how they manifest in a production environment. This paper presents our experience tackling these challenges at Facebook, which uses dozens of PoPs to serve two billion users across the vast majority of countries. We make three primary contributions.

First, since the impact of the above-mentioned limitations of BGP depends on a network's design, we describe the network connectivity and traffic characteristics that make it challenging for providers of popular applications to manage their egress traffic. At Facebook, it is common for PoPs to have four or more routes to many client networks. Although many interconnections in the US have spare capacity [8], in our measurements across 20 PoPs, 10% of egress interfaces experience a period in which Facebook's BGP policy would lead to BGP assigning twice as much traffic as the interface's capacity! Moreover, the traffic demand from a PoP to a prefix can be unpredictable, with traffic rates to the same prefix at a given time exhibiting as much as 170x difference across weeks. Thus, while Facebook's rich connectivity provides shorter paths, more options for routing, and significant capacity in aggregate, the capacity constraints of individual paths and irregular traffic makes it difficult to use this connectivity. Traffic must be dynamically routed in order to optimize efficiency and performance without exceeding these constraints.

Second, we present the design of Edge Fabric, a system for optimized routing of egress traffic. Edge Fabric receives BGP routes from peering routers, monitors capacities and demand for outgoing traffic, and determines how to assign traffic to routes. Edge Fabric enacts its route selections by injecting them (using BGP) into the peering routers, overriding the router's normal BGP selection. Edge Fabric has been deployed in production for over four years. We evaluate how Edge Fabric operates in production and share how Edge Fabric's design has evolved over time. In addition, we discuss technical challenges faced at Facebook's scale and lessons learned.

Third, we instrument Edge Fabric to continually measure performance along alternate paths—not just its best choice—to every prefix. Edge Fabric gathers these measurements by routing a small fraction of the traffic to every prefix along alternate paths. Our measurements from 4 PoPs show that 5% of prefixes could see a reduction in median latency of 20+ms by choosing an alternative to BGP's prefered route.

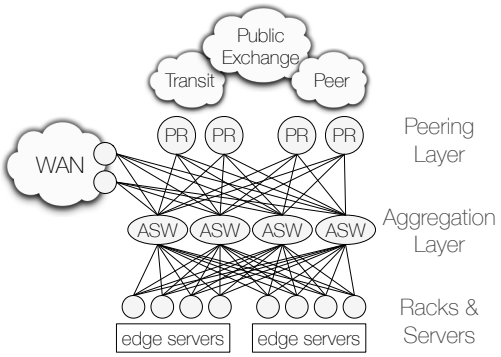

Figure 1: A PoP has Peering Routers, Aggregation SWitches, and servers. A private WAN connects to datacenters and other PoPs.

2 SETTING

2.1 Points of Presence

To help reduce user latencies, Facebook has deployed Points of Presence (PoPs) in dozens of locations globally. A PoP serves users from racks of servers, which connect via intermediate aggregation switches (ASWs) to multiple peering routers (PRs), as seen in Figure 1. ASWs maintain BGP sessions with PRs and rack switches. Each PoP includes multiple peering routers (PRs) which exchange BGP routes and traffic with other ASes. The use of multiple PoPs reduces latencies in two ways: 1) they cache content to serve users directly, and 2) when a user needs to communicate with a data center, the user's TCP connection terminates at the PoP which maintains separate connections with data centers, yielding the benefits of split TCP [9] and TLS termination.

Only a single PoP announces each most-specific Facebook prefix, and so the PoP at which traffic ingresses into Facebook 's network depends only on the destination IP address.[1] A global load balancing system assigns users to PoPs via DNS (returning an IP address of a particular PoP in response to a DNS request for a general hostname) and by injecting PoP-specific URLs into responses (that resolve only to IP addresses at a particular PoP). Like similar systems [4], it injects measurements to capture the performance to users from alternate PoPs. The load balancing system uses these measurements to direct users to their "best" performing PoPs (subject to constraints such as capacity; PoPs currently not serving users for maintenance, troubleshooting, or testing; and agreements with other networks). For 80% of user networks, it directs all requests from the network to a single (nearby) PoP (during a day in Jan. 2017). Details of the load balancing are out of the paper's scope, and we design Edge Fabric assuming it has no control over which PoP serves a given user.

Figure 2 depicts the relative volume traffic served from 20 PoPs, a subset selected for geographic and connectivity diversity that combined serve most Facebook traffic. The paper refers to the PoPs consistently by number, ordered by volume. At these PoPs, 95% of the traffic comes from clients in $\approx 65,000$ prefixes (during a day in Jan. 2017). Considering just the client prefixes needed to account for 95% of a PoP's traffic, Figure 3 shows that each PoP

[1] A covering prefix announced across PoPs guards against blackholes if a more-specific route fails to propagate to a router.

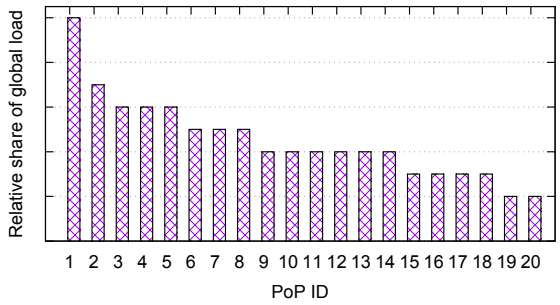

Figure 2: Relative egress traffic volume (rounded) of 20 PoPs.

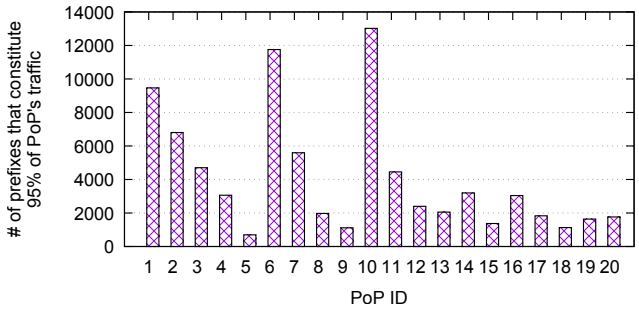

Figure 3: # of BGP prefixes to constitute 95% of PoP's traffic.

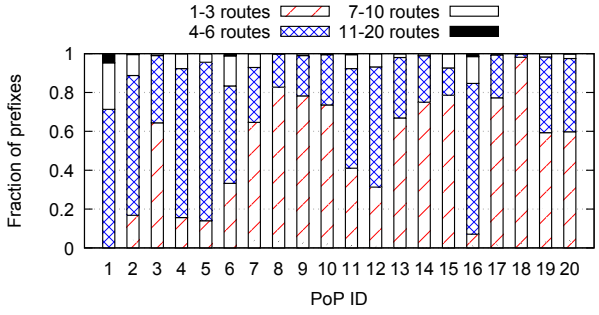

Figure 4: # routes to prefixes contributing 95% of PoP's traffic.

serves ≈ 700 to ≈ 13,000 prefixes, and 16 PoPs send 95% of their traffic to fewer than 6500 prefixes.

2.2 Interdomain Connectivity

PRs' BGP connections to other ASes are of different types:

- *Transit providers* provide routes to all prefixes via a private network interconnect (PNI) with dedicated capacity just for traffic between Facebook and the provider.
- *Peers* provide routes to the peer's prefixes and to prefixes in its customer cone [18]. Peers vary by connection type:
  - *Private peers*: PR connects to peer via a dedicated PNI.
  - *Public peers*: PR's BGP session to peer and traffic to peer traverse the shared fabric of an Internet exchange point (IXP).
  - *Route server peers*: PR receives peer's routes indirectly via a route server [2] and exchanges traffic across the IXP fabric.

A PoP may maintain multiple BGP peering sessions with the same AS (e.g., a private peering and a public peering, or at multiple PRs). Most PoPs connect to 2+ transit providers, with each transit provider maintaining a BGP session with 2+ of the PoP's PRs, for capacity and failure resilience. When possible, all PRs maintain equal PNI capacity to a given peer, although sometimes some PRs have different capacity or do not connect to the peer at all.

PoP ID	1 (EU)		2 (AS)		11 (EU)		16 (AS)		19 (NA)	
	Peers	Traffic	Pe.	Tr.	Pe.	Tr.	Pe.	Tr.	Pe.	Tr.
Private	.12	.59	.25	.87	.02	.24	.21	.78	.13	.73
Public	.77	.23	.39	.04	.45	.45	.54	.13	.85	.07
Rt Srvr	.10	—	.34	—	.52	—	.23	—	0	—
Transit	.01	.18	.01	.10	.01	.31	.02	.08	.01	.20

Table 1: Fraction of peers and of traffic to peers of various types at example PoPs in EUrope, ASia, and North America. A peer with both a private and a public connection will count in both. A peer with a public and a route server connection counts as public (they share an IXP port). Traffic to public and route server peers is combined.

In general, we configured Facebook's network to egress a flow only at the PoP that the flow enters at, rather than routing across the WAN from servers in one PoP to egress links at a different PoP. Isolating traffic within a PoP reduces backbone utilization, simplifies routing decisions, and improves system stability (§8.1.3). Even with this simplification, Facebook has diverse routing options. Figure 4 shows the distribution of the number of routes that each PoP could choose from to reach the prefixes that make up 95% of its traffic. If a peer provides the same path through a public peering and a route server, or if multiple PRs receive the same route from the same peer, we only count it once. Although not derivable from the graph (which combines destinations with 1-3 routes), all PoPs except one have at least two routes to every destination, and many have four or more routes to most prefixes.

We configure PRs to prefer peer routes to transit routes (via `local_pref`), with AS path length as a tiebreaker. When paths remain tied, PRs prefer paths from the following sources in order: private peers > public peers > route servers.[2] We encode peer type in MEDs (and strip MEDs set by the peer, which normally express the peer's preference of peering points but are irrelevant given that Facebook egresses a flow at the PoP where it ingresses). The preference of peers over transit recognizes that an AS that peers with Facebook expects to receive its traffic on that link. In addition, we have found that peer routes frequently have better performance and a lower risk of downstream congestion. Short AS paths may be more direct or give the traffic to the destination AS sooner [5]. By preferring the dedicated capacity of a private peering over a connection across a shared IXP fabric, our policy avoids the possibility of cross-congestion at the egress and respects that the peer dedicated resources to receiving Facebook traffic.

We configured BGP at PRs and ASWs to use BGP multipath. When a PR or an ASW has multiple equivalent BGP best paths for the same destination prefix (as determined by the BGP best path selection algorithm), it distributes traffic across the equivalent routes using Equal-cost multi-path routing (ECMP).

Overall, Facebook has thousands of peer ASes. Table 1 shows, for example PoPs, the fraction of peers that are of each type. Each PoP shown has hundreds of peers in total, yielding rich connectivity. The table also shows the fraction of its traffic that each PoP can serve

[2] We de-prioritize a handful of private peers relative to public peers for policy reasons, but the effect is minor in the context of this paper.

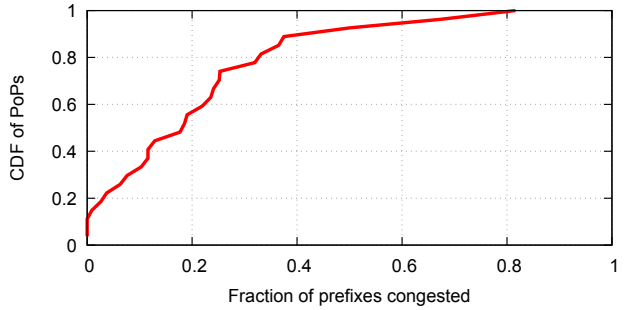

Figure 5: Distribution across PoPs of fraction of prefixes that would have experienced congestion had Edge Fabric not intervened.

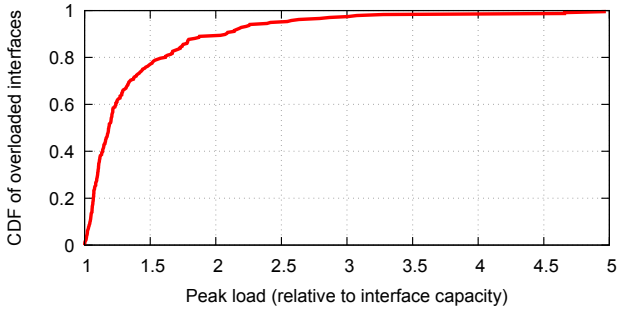

Figure 6: Distribution across interfaces of ratio of *peak load* to *capacity* (including only interfaces that would have experienced congestion had Edge Fabric not intervened).

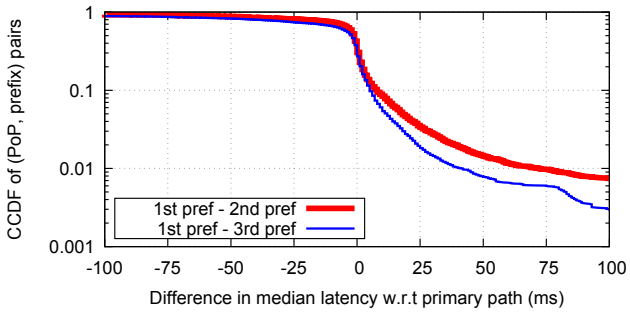

Figure 7: Latency on BGP's preferred path to a prefix vs. on alternate paths to the prefix (data from four PoPs).

by peer type (assuming all traffic is assigned to its most preferred route without considering capacity). Although private peers make up at most a quarter of peers at any PoP, they receive the majority of traffic at all but PoP-11. It is typical to pursue the dedicated capacity of private interconnects for high-volume peerings. At all but PoP-11, 80+% of traffic egresses to private, public, and route server peers rather than transit, an example of how today's large providers "flatten" the Internet [5, 15]. However, the distribution of peer types varies widely across PoPs by count and by traffic.

3 CHALLENGES OF BGP

As demand increased and Facebook rapidly expanded its PoP infrastructure and connectivity, we encountered challenges due to limitations of BGP, leading us to build Edge Fabric. Any static interdomain routing policy will likely suffer similar challenges.

Peering capacity is limited, but BGP is not capacity-aware. Although Facebook builds PoPs, expands capacity, and pursues private interconnections, a link's capacity may not suffice to deliver all traffic that Facebook would like to send over it. Rapid growth in demand can quickly make the capacity of an existing interconnection insufficient, and augmenting capacity in some cases is impossible or can take months. Short-term spikes in demand (perhaps due to an event or a holiday) or reductions in capacity due to failures cause volatility in demand and available capacity. Further, PoPs serve nearby users, and so diurnal patterns can lead to synchronized peaks in demand, causing very high utilization that can exceed PNI capacity for short periods. In addition, capacity to a given peer may be unequally distributed across PRs, but ECMP at ASWs will be unaware of this imbalance and will evenly distribute traffic across PRs, which can result in overload at some PRs and poor utilization of capacity at others (Section 8.1.4 describes why we do not use weighted-cost multipath). In general, assigning more traffic to an egress interface than it (or the downstream path) can handle causes congestion delay and packet loss, and it also increases server utilization (due to retransmissions) [10].

This paper presents our system to enable capacity-aware egress decisions on top of BGP. To understand the scale of the problem, we analyzed a two-day log from January 2017 of each prefix's per-PoP egress traffic rate (averaged over a 1 minute window) and compared the capacity of Facebook's egress links to the rate of traffic that BGP would assign to them (based on our configured BGP policy from §2.2), if Edge Fabric did not intervene to prevent overload.

For each PoP, Figure 5 shows the fraction of prefixes that would experience congestion. Most PoPs are capacity-constrained for at least one prefix, and a small fraction of PoPs are capacity-constrained for most prefixes. For any interface overloaded at least once, Figure 6 shows peak load per interface (computed over 1-minute intervals, relative to interface capacity). While most interfaces experience low amounts of overload (median load = 1.19X capacity), 10% experience a period in which BGP policy would lead to BGP assigning twice as much traffic as the interface's capacity! In many cases, Facebook has found that it cannot acquire sufficient additional capacity to the peer, motivating Facebook to build Edge Fabric to overcome BGP's inability to handle this overload on its own.

BGP decisions can hurt performance. Facebook's BGP policy favors paths that are likely to optimize network traffic performance. The policy avoids transit routes when peer routes (which are often better) exist, tiebreaks in favor of short AS paths (usually lower latency), and prefers peers with dedicated peering capacity over public peers which may encounter cross-congestion. However, BGP itself is not performance-aware, and so this policy relies on attributes such as AS path length that serve as imperfect heuristics for proximity and performance.

To understand the severity of this problem, we compare the performance of alternative paths at four PoPs, one in North America (PoP-19 in §2), one in Europe (PoP-11), and two in Asia Pacific (PoPs-2, 16). We shift a fraction of actual user traffic for all IP prefixes onto the second and third best BGP paths. We describe how we conduct these measurements in more detail in Section 6.2. Figure 7 depicts

the difference in median round-trip latency between the preferred and less preferred paths. The figure shows that 5% of <PoP,prefix> pairs could see an *improvement* of 20+ms (a threshold Facebook considers significant) if switched from BGP's preferred path to its second preference, and 3% could see such an improvement if switched to BGP's third preference. At the scale of Facebook, this relatively small fraction of prefixes represents a lot of traffic. The results also show that the second preference is within 20ms of the preferred path for 74% of <PoP,prefix> pairs, suggesting that we may be able to avoid overload by detouring traffic to less-preferred routes, without resorting to paths that have much worse baseline performance.

4 GOALS AND DESIGN DECISIONS

Our goal is to overcome BGP's limitations (§3) and enable Facebook to use its interdomain connectivity (§2.2) to improve performance:

- Given limited capacity and the performance impact of congestion, routing decisions must be aware of capacity, utilization, and demand.
- Decisions should also consider performance information and changes, while simultaneously respecting policies.

Towards these goals, in 2013 we began building Edge Fabric, a traffic engineering system that manages egress traffic for our PoPs worldwide. Section 5 describes Edge Fabric's current approach to automatically shift traffic to avoid overloading links. Section 8.1 describes how we have evolved Edge Fabric over time, due to changing needs and thanks to operational experience that let us improve its stability and ability to combat congestion. In addition, we have studied how we can use measurements to improve egress routing performance, and Section 6 discusses how we have begun to incorporate them into our routing decisions.

We now present main design decisions of Edge Fabric.

Operate on a per-PoP basis. While Edge Fabric assigns traffic to egress routes, the global load balancing system maps a user request to ingress at a particular PoP (§2.1), and a flow egresses at the same PoP at which it ingresses. So, Edge Fabric need only operate at a per-PoP granularity; it does not attempt to orchestrate global egress traffic. This design allows us to colocate its components in the PoP, reducing dependencies on remote systems and decreasing the scope and complexity of its decision process. We can then restart or reconfigure Edge Fabric at a PoP in isolation (§5.4) without impacting other PoPs (outside of the load balancing system directing them more traffic).

Centralize control with SDN. We chose to use an SDN-based approach, in which a centralized controller receives network state and then programs network routing decisions. This approach brings benefits of SDN: it is easier to develop, test, and iterate compared to distributed approaches. Because Facebook connects to its peers using BGP, part of the network state is the BGP paths Facebook receives, which are continuously streamed to the controller (§5.1.1).

Incorporate real-time traffic and performance measurements into decisions. The controller receives measurements of capacity and demand multiple times per minute (§5.1.2), enabling Edge Fabric to maximize utilization of preferred paths without overloading them (§5.2). Facebook has existing server-side monitoring of client performance (§6.2.3). We added the ability to measure multiple paths to a destination prefix in parallel by routing a fraction of production flows (selected at random, §6.2.1) using alternate routing tables at the PRs (§6.1 and §6.2.2). This approach guarantees that measurements capture user-perceived performance. Edge Fabric can identify cases where BGP's preferred routing is not optimal, laying the groundwork for performance-based decisions.

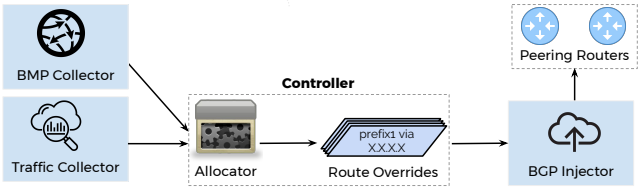

Figure 8: Edge Fabric components

Use BGP for both routing and control. Despite the centralized controller, every PR makes local BGP route decisions and PRs exchange routes in an iBGP mesh; the controller only intervenes when it wants to override default BGP decisions. To override a decision, Edge Fabric sets its preferred route to have high `local_pref` and announces it via BGP sessions to PRs, which prefer it based on `local_pref` (§5.3). Building Edge Fabric atop our established BGP routing simplifies deployment, lets the network fall back to BGP for fault tolerance, and leverages existing operational teams, their expertise, and network monitoring infrastructure.

Leverage existing vendor software and hardware. We use battle-tested vendor gear and industry standards, avoiding the need for custom hardware or clean slate design. Sections 5 and 6 describe our use of BGP, IPFIX, sFlow, ISIS-SR, eBPF, and BMP, and Section 8.1 explains how specifics of vendor support have influenced our design.

Overall, Edge Fabric's design values simplicity and compatibility with existing infrastructure, systems, and practices. Its approach to satisfy our primary goal—avoid overloading egress interfaces (§5)—does not require any changes to our servers or (browser-based or app-based) clients, adding only BGP sessions between the routers and the controller. Our secondary goal of enabling performance-aware routing (§6) relies on straightforward software changes at servers and the addition of alternate routing tables at routers, standard functionality supported by our existing equipment.

5 AVOIDING A CONGESTED EDGE

Edge Fabric consists of loosely coupled microservices (Figure 8). Every 30 seconds, by default, the allocator receives the network's current routes and traffic from other services (§5.1), projects interface utilization (§5.1.2), and generates a set of prefixes to shift from overloaded interfaces and for each prefix, the detour path to shift it to (§5.2). Another service enacts these overrides by injecting routes into routers via BGP (§5.3). We use a 30 second period to make it easier to analyze the controller's behavior, but we can lower the period if required due to traffic volatility.

5.1 Capturing Network State (inputs)

Edge Fabric needs to know all routes from a PoP to a destination, and which routes traffic will traverse if it does not intervene. In addition, it needs to know the volume of traffic per destination prefix and the capacity of egress interfaces in the PoP.

5.1.1 Routing information.

All available routes per prefix. The BGP Monitoring Protocol (BMP) allows a router to share a snapshot of the routes received from BGP peers (e.g., all of the routes in its route information base, or RIB) and stream subsequent updates to a subscriber [25]. The BMP collector service maintains BMP subscriptions to all peering routers, providing Edge Fabric with a live view of every peering router's RIB. In comparison, if Edge Fabric maintained a BGP peering with each router, it could only see each router's *best* path.[3]

Preferred paths per prefix. BMP does not indicate which path(s) BGP has selected, and a BGP peering only shares a single path, even if the router is using ECMP to split traffic across multiple equivalent paths. The controller also needs to know what paths(s) would be preferred without any existing overrides. So, the controller emulates BGP best path selection (including multipath computation and ignoring existing overrides that would otherwise be preferred) for every prefix.

5.1.2 Traffic information.

Current traffic rate per prefix. The Traffic Collector service collects traffic samples reported by all peering routers in a PoP (using IPFIX or sFlow, depending on the router), groups samples by the longest-matching prefix announced by BGP peers, and calculates the average traffic rate for each prefix over a two-minute window. We use live rather than historical information because, for example, the global load balancing system (§2) may have shifted traffic to/from the PoP, destination networks may have changed how they originate their network space for ingress traffic engineering, and traffic demands change over time on a range of timescales.

If the rate of a prefix exceeds a configurable threshold, for example 250 Mbps, the service will recursively split the prefix (e.g., splitting a /20 into two /21s, discarding prefixes with no traffic) until the rate of all prefixes is less than the threshold. Splitting large prefixes allows the allocator to make more fine-grained decisions and minimize the amount of traffic that must be detoured when interfaces are overloaded (§5.2).

Interface information. The allocator retrieves the list of interfaces at each peering router from a network management service [27] and queries peering routers via SNMP every 6 seconds to retrieve interface capacities, allowing the allocator to quickly adapt to capacity changes caused by failures or provisioning.

Projecting interface utilization. The allocator projects what the utilization of all egress interfaces in the PoP would be if no overrides had been injected, assigning each prefix to its preferred route(s) from its emulated BGP path selection. The BGP best path computation process may return multiple (equivalent) routes. These routes may be spread across multiple interfaces and/or peering routers. In these cases, the allocator assumes that ECMP at both the aggregation layer and peering routers splits traffic equally across the paths.

We *project* interface utilization instead of using the *actual* utilization to enable the allocation process to be *stateless*, simplifying our design: the allocator generates a full allocation from scratch on each cycle and does not need to be aware of its previous decisions or their impact. Section 8.1.1 discusses this design choice in detail.

Based on the projected utilization, the allocator identifies interfaces that will be overloaded if it does not apply overrides. We consider an interface overloaded if utilization exceeds ~95% (the exact threshold can vary based on interface capacity and peer type), striking a balance between efficient utilization and headroom to handle volatility (including microbursts).

5.2 Generating Overrides (decisions)

The allocator generates overrides to shift traffic away from interfaces that it projects will otherwise be overloaded. For each overloaded interface, the allocator identifies the prefixes projected to traverse the interface and, for each prefix, the available alternate routes.[4] It then identifies the single ⟨prefix, alternate route⟩ pairing that it prefers to shift first from the interface, applying the following rules in order until a single preference emerges:

1. Prefer IPv4 over IPv6 prefixes.[5]
2. Prefer prefixes that a peer prefers Facebook detour. Peers have the option of providing these preferences using Facebook-defined BGP communities.
3. Among multiple alternate routes for a given prefix, prefer routes with the longest prefix.[6]
4. Prefer paths based on BGP's best path selection process. For instance, the allocator will prefer shifting a prefix with an available route via a public exchange over a prefix that only has an alternate route via a transit provider (§2.2).
5. Prefer paths based on an arbitrary but deterministic tiebreaker. The tiebreaker selects first based on the prefix value. If there are equally preferred alternate routes for the chosen prefix, the allocator orders alternate routes in a consistent way that increases the likelihood of detour traffic being balanced across interfaces.

Once a pairing has been selected, the allocator records the decision and updates its projections, removing the prefix's traffic from the original interfaces and placing all of the PoP's traffic for the prefix onto the selected alternate route's interface. Edge Fabric detours all traffic for the prefix, even if the prefix's primary route was across multiple interfaces or routers. However, the total traffic per prefix is always less than the threshold that Traffic Collector uses when splitting high traffic prefixes (§5.1).

[3] Previously, we used BGP add-path capability to collect multiple routes from each router, but some vendor equipment limits the number of additional paths exchanged, which we quickly exceeded given our rich interdomain connectivity.

[4] A prefix will not have any alternate routes if all routes to it are on interfaces that lack sufficient spare capacity (after accounting for earlier detours from this round of allocation). This scenario is rare, as transit interfaces have routes to all prefixes and, in our evaluation, always had at least 45% of their capacity free (§7.2).

[5] We prefer shifting IPv4 prefixes because we have experienced routes that blackhole IPv6 traffic despite advertising the prefix. If Edge Fabric shifts traffic to such a route, end-users will fallback to IPv4 [30], causing traffic to oscillate between IPv4 and IPv6.

[6] Unlike the standard BGP decision process, the allocator will consider using routes for less-specific prefixes, just with lower preference.

The allocator continues to select prefixes to shift until it projects that the interface is no longer overloaded or until the remaining prefixes have no available alternate routes.

Because the allocation process is stateless, it generates a new allocation from scratch every 30 seconds. To minimize churn, we implemented the preferences to consider interfaces, prefixes, and detour routes in a consistent order, leading the allocator to make similar decisions in adjacent rounds. The remaining churn is often due to changes in traffic rates and available routes.

The headroom left by our utilization thresholds allows interface utilization to continue to grow between allocation cycles without interfaces becoming overloaded. If a route used by the controller for a detour is withdrawn, the controller will stop detouring traffic to the withdrawn route to prevent blackholing of traffic.

5.3 Enacting Allocator Overrides (output)

In each round, the allocator generates a set of BGP updates for Edge Fabric's overrides and assigns each update a very high `local_pref`. The allocator passes the BGP updates to the BGP Injector service, which maintains a BGP connection with every peering router in the PoP and enacts the overrides by announcing the BGP updates to the target routers. Because the injected updates have a very high `local_pref` and are propagated between PRs and the ASWs via iBGP, all routers prefer the injected route for each overridden prefix. The injector service then withdraws any overrides that are no longer valid in the current allocation.

We configure Edge Fabric and the global load balancer such that their independent decisions work together rather than at odds. First we need to protect against Edge Fabric decisions and global load balancer decisions interacting in ways that cause oscillations. In selecting which PoP to direct a client to, the global load balancer jointly considers performance from the PoP and Facebook's BGP policy's preference for the best route from the PoP, but we configure it to ignore routes injected by Edge Fabric and instead to consider the route that would be used in the absence of an override. If the load balancer was allowed to consider the override route, it could shift client traffic away from a PoP in reaction to Edge Fabric detouring traffic from an overloaded interface to a less-preferred route. This shift would reduce traffic at the PoP, lowering Edge Fabric's projection of interface load, which could cause it to stop detouring, opening the possibility of an oscillation. Second, the global load balancer can track interface utilization and appropriately spread traffic for a client network across all PoPs that it prefers equally for that network. So, Edge Fabric need only intervene with overrides once interfaces are overloaded across all the PoPs.

5.4 Deploying, Testing, and Monitoring

We typically deploy Edge Fabric controller updates weekly, using a multi-stage release process to reduce risk. First, because our design is modular and stateless, we write comprehensive automated tests for individual components of Edge Fabric and Edge Fabric's dependencies. Second, because our controller is stateless and uses *projected* interface utilization instead of *actual* utilization, we can run a *shadow* controller inside a sandbox that can query for the same network state as the live controller, without needing any state from the live controller and without being dependent on its earlier decisions. We continuously run *shadow* instances, built from the latest revision, for every PoP and compare the decisions and performance of shadow instances against the controllers running in production. We review these comparisons before beginning deployment of a new version. Third, because we deploy Edge Fabric and all dependencies on a per-PoP basis, we can roll out new versions of a controller and its dependencies on a PoP-by-PoP basis (an automated system performs this). While the Edge Fabric controller is being updated, the BGP Injector service continues to inject the previous round of decisions until the controller resumes (a process that takes less than 5 minutes). If we need to update the BGP Injector service, hold timers at PRs maintain existing injections until the injector has restarted.

While the stateless controller is amenable to automated tests, it is particularly vulnerable to errors in BGP route or traffic rate data, as these can cause the controller to misproject interface utilization. To catch misprojections, a monitor compares the controller's post-allocation projection of interface utilization with actual interface utilization, and it raises an alarm if they differ by 5% for more than a configurable period of time. Through this process, we identified and corrected bugs in our routing policy and in how our PRs export IPFIX and sFlow samples. The controller projects that ECMP will distribute traffic nearly evenly across links. The monitor identified instances of ECMP unexpectedly distributing traffic in a highly-unbalanced manner, which Edge Fabric can mitigate by overriding the multipath to send traffic to a single PR.

Similarly, while Edge Fabric's use of BGP and distributed route computation lets us build on existing infrastructure, it also exposes us to the underlying complexity of the BGP protocol. In one scenario, a configuration issue caused routes injected by Edge Fabric to not be reflected across the full PoP, shifted traffic away from the previously overloaded interface, but to a different interface than desired. To detect such misconfigurations, we built an auditing system that regularly compares Edge Fabric's output against current network state and traffic patterns.

6 TOWARDS PERFORMANCE-AWARE ROUTING

Edge Fabric avoids performance problems due to congested links at the edge of Facebook's network. Yet, Facebook's users can still suffer sub-optimal performance because of the limitations of the BGP decision process (§3). BGP may not choose the best-performing path for a given prefix's default path, and likewise, when detouring a prefix, BGP may not choose the best-performing detour path. Even if BGP does choose the best-performing path in all scenarios, performance can be degraded when a prefix is detoured if its best detour path has worse performance than its default path.

Moreover, while Facebook connects directly to many edge networks (like other large content providers do [5]), performance to other edge networks (to which Facebook routes traffic via IXPs or transit networks) can be hindered by downstream congestion, which Edge Fabric (as described in §5) does not account for. Even in cases when Facebook does directly connect to the edge network, downstream congestion can still degrade performance. Due to transient failures and volatility in traffic and congestion, which path performs *best* can vary over time, and so performance-based decisions need to be responsive to change. BGP provides neither

visibility into performance nor the explicit ability to make decisions based on it.

To enable routing decisions to incorporate performance, we need to measure multiple paths to a destination in parallel, continuously. In addition, to enable Edge Fabric to best utilize available capacity, we need to be able to prioritize certain types of content (for instance, prioritizing a live video stream when deciding which traffic to place on an egress interface where demand exceeds capacity). However, because BGP only supports destination-based routing, BGP alone cannot realize either of these objectives.

We first describe how we direct specific flows onto specific paths (§6.1), then describe how we use that capability to measure the performance of multiple paths to a prefix in parallel (§6.2). We have deployed these two mechanisms in production alongside Edge Fabric (§5). We finish this section by discussing future use cases that employ those measurements for directing specific flows to enable performance-aware routing (§6.3).

6.1 Placing Traffic on Alternate Paths

To sidestep BGP's limitation that it only supports destination-based routing, we build a mechanism that allows us to route specific flows via select paths. The mechanism requires only minimal modifications at servers and peering routers, and does not require coordination between servers and other network elements: servers can make decisions on a per-flow basis without having any knowledge of network state at peering routers.[7]

- **Servers select and mark flows** that should be treated specially. Specifically, servers set the DSCP field in the IP packets of selected flows to one of a set of pre-defined values. Servers can mark a flow to, for instance, signal that it can be used to measure the performance of an alternate path or to signal that the flow is carrying real-time traffic and is performance-sensitive.
- **Policy routing at the PRs** match on the markings applied by servers and route corresponding packets based on alternate routing tables. We install a unique routing table for each DSCP value.
- **A controller injects routes** into the alternate routing tables at PRs to control how marked flows are routed. If the controller has not injected a route for a particular destination, marked flows will be routed based on the PR's default routing table.

This approach does not require continued synchronization between servers, routers, and controllers. Servers can continuously tag packets without any knowledge of network state, and controllers can inject routes into alternate routing tables to control the route of marked packets only as needed.

In addition, the aggregation switch (ASW) layer need not be aware of DSCP values assigned to flows and can blindly forward IP traffic based on BGP's best route for the destination prefix. However, a PR with the best route may not connect to the peer providing the alternate route, since different PRs at a PoP can have different peers. To handle this case, the controller injects the alternate route at PRs that lack the peer, setting the nexthop of the route to be a PR that connects to the peer. Trying to forward traffic from one PR to another via IP would cause a loop since the aggregation layer forwards traffic based on BGP's best route for the destination. To avoid this, we configure the PRs to address each other via labels (using ISIS-SR) and tunnel traffic via ASWs using MPLS forwarding.

[7] Per-flow overrides avoid out-of-order packets that slow TCP.

6.2 Measuring Performance of Alternate Paths

End-user TCP connections terminate at front-end servers at the edge of our network, which in turn proxy HTTP requests to backend servers [26]. We use the mechanism described in Section 6.1 to randomly select a subset of these connections (§6.2.1) and route them via alternate paths (§6.2.2). This allows us to collect measurements with Facebook's existing infrastructure that logs the performance of client connections observed by the front-end servers (§6.2.3).

6.2.1 Randomly selecting flows. We implemented a program that runs on front-end servers to randomly select flows and mark them to be routed via an alternate path using the Extended Berkeley Packet Filter (eBPF) instruction set. eBPF allows the program to be loaded into the kernel where it can efficiently process all packets egressing from the server. With this approach, no changes are required to existing client or server applications.

The eBPF program randomly selects a configurable fraction of flows for alternate path measurements and then sets the DSCP field in all of the flow's IP packets to a DSCP value reserved for alternate path measurements. The program's configuration can be dynamically updated and contains the percentage of flows that should be randomly mapped to each DSCP value reserved for measurements. For instance, to measure two extra paths per prefix, the configuration could assign 0.75% of flows the DSCP value *12* and 0.25% of flows the DSCP value *24*. Given the scale of Facebook, sampling a small fraction of connections still results in a large set of measurements. By using passive measurements based on existing production traffic, we avoid active measurements (e.g., pings) which may not represent performance as perceived by real users.

6.2.2 Injecting routes. We built an AltPath controller to inject routes into alternate routing tables. The controller only generates alternate paths for prefixes with traffic in the past 10 minutes, drastically reducing the size of alternate routing tables (§2.2).

Every 30 seconds, the AltPath controller uses BGP routes retrieved from the BMP Collector service (§5.1) to decide on an alternate path for each prefix, for each of the DSCP values being assigned by servers. The controller then uses the BGP Injector service (§5.3) to inject alternate routes for each DSCP value into the corresponding routing table at PRs.

The AltPath controller takes as input a set of destination ASNs that it will not perform alternate path measurements for. Our traffic engineering team adds to the list of networks that are known (from prior operational experience) to have extremely poor performance on alternate paths to avoid negatively impacting end-users.

6.2.3 Measuring performance. At the termination of (a sample of) client TCP connections, our front-end servers log metrics that we use to evaluate path performance. These servers use a separate eBPF program to capture statistics at connection termination for sampled TCP connections, with a typical sampling rate of 1 out of every 1000 connections. The statistics include retransmission rate, retransmission timeout (RTO), smoothed round-trip times (SRTT), and the number of segments sent/received. In addition, the servers sample HTTP transactions and measure per-response client

download goodput. The servers also record the value assigned to the connection's DSCP field by the eBPF program, if any. A collector joins samples with egress route information so that samples can be grouped by the route they traversed and performance statistics can be aggregated per route.

6.3 Future Uses of Performance-Aware Routing

The mechanisms described in Section 6.1 and Section 6.2 allow us to route traffic along alternate paths and measure performance. In this section, we explore how we can build atop these mechanisms to improve user performance.

6.3.1 Overriding paths chosen by BGP. Today, Edge Fabric overrides BGP's default decision process to mitigate congestion at the edge of Facebook's network. Going forward, we can improve Edge Fabric by incorporating AltPath measurements. First, Edge Fabric can use the measurements to identify scenarios where it may be possible to improve performance by overriding BGP's default decisions, even though the edge is not congested. When performance on the current path degrades, AltPath measurements can identify whether the problem can be avoided via a path change, or whether the problem impacts all paths (as might happen with severe congestion close to the destination). Second, Edge Fabric can use alternate path measurements to ensure that traffic is placed onto the best performing *detour* path whenever congestion does occur. We provide initial results on the promise of these use-cases in Section 7.3. Finally, alternate path measurements can help network operators identify how performance will be impacted if traffic is detoured, which can be used in network planning decisions.

6.3.2 Optimizing use of limited capacity. When capacity is limited, Edge Fabric may be forced to detour traffic to paths with comparatively worse performance. Edge Fabric can be extended to best use the limited capacity on the primary path by shifting prefixes and/or flows that measurements indicate are less likely to be impacted. First, Edge Fabric can amend its decision criteria (§5.2) to prefer shifting prefixes that will experience little performance degradation on their detour path. Second, higher priority flows can be routed over the constrained path. Front-end servers can assign predefined DSCP values to flows that are higher priority, such as a live video stream. Then, when Edge Fabric injects routes to shift default-routed traffic away from an overloaded interface, it can in parallel inject routes into an alternate routing table to keep flows that are marked with a high priority DSCP value on the better performing path. Both IPFIX and sFlow samples collected by Traffic Collector include the DSCP field, thereby making it possible for Edge Fabric to determine the rate of traffic marked with a given DSCP value and account for this in its projection.

7 RESULTS ON PRODUCTION TRAFFIC

7.1 Deployment Status and Evaluation Datasets

We deployed Edge Fabric for all production traffic, detouring traffic to avoid overloading interfaces at PoPs around the world. Section 7.2 describes a two-day study in January 2017, predating our current stateless controller. This study used our earlier stateful controller, which also did not automatically split large-volume prefixes. We believe that our stateless controller achieves better utilization than the stateful one (without negative side effects) but have not yet formally evaluated it.

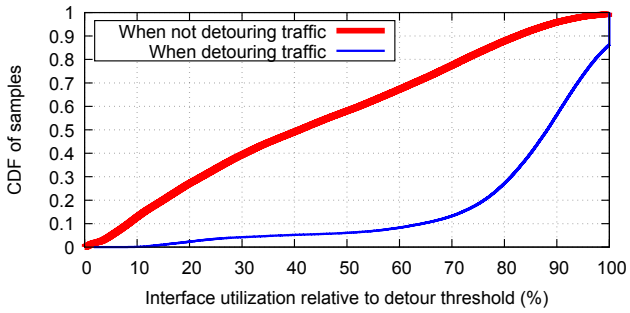

Figure 9: Utilization of interfaces relative to detour thresholds.

We are now deploying AltPath across our PoPs, but the results in Section 7.3 are from our initial trial deployment at four PoPs beginning in late 2016, chosen in part for their rich connectivity: one in North America (PoP-19 in §2), one in Europe (PoP-11), and two in Asia Pacific (PoPs-2, 16). Combined, these PoPs account for approximately 18% of the total traffic of the 20 PoPs. We do not currently use AltPath measurements automatically to inform production routing decisions, but Section 7.3 presents trial results on the impact and challenges of integrating performance into routing decisions.

We created two alternate path tables at each router, populated with BGP's 2nd and 3rd preferred paths for all prefixes. Our 2016 measurements predate our current DSCP-based approach and instead assign flows to tables based on the flows' destination ports. For each alternate routing tables, we generated a distinct set of ports that matches approximately 0.5% of the total traffic, and then installed rules at PRs to route traffic destined towards these ports to one of the tables. To increase the number of measurements of alternate paths, we apply a 100x multiplier to the existing measurement sampling rate whenever a connection uses a port in the AltPath port set. With this multiplier, each alternate path receives 0.5% of traffic and has ≈ 50% as many measurements as the primary path.

7.2 Evaluating Capacity-Aware Routing

Does Edge Fabric achieve its primary goal, preventing congestion at edge interfaces while enabling efficient utilization? Edge Fabric prevents congestion by detouring traffic to alternate routes. During the study, non-overloaded alternate routes always existed, giving Edge Fabric options to avoid overloading interfaces. In particular, transit providers can take detoured traffic to any destination, and the maximum instantaneous transit utilization observed at any individual PoP (sampled at one minute intervals) during the study was 55%. Edge Fabric successfully prevented egress traffic from overloading egress interfaces, with no packet drops at an interface when Edge Fabric was not detouring traffic from it, nor in 99.9% of periods in which it was detouring. Figure 9 shows the utilization on these interfaces (relative to their detour thresholds) during these periods; Edge Fabric keeps utilization of preferred routes high even while avoiding drops, and utilization is below a safe threshold during periods in which Edge Fabric decides not to detour traffic.

How much traffic does Edge Fabric detour? Figure 10 shows the distribution of the fraction of time that Edge Fabric detoured traffic

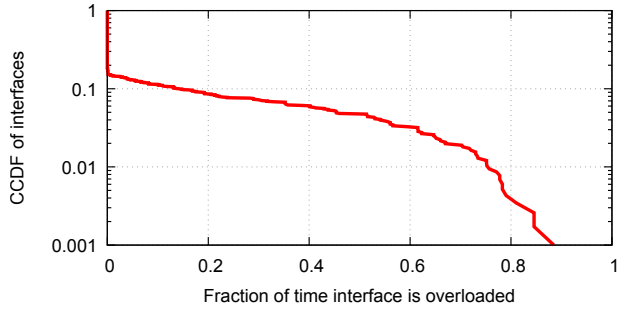

Figure 10: Fraction of time Edge Fabric detours from interfaces.

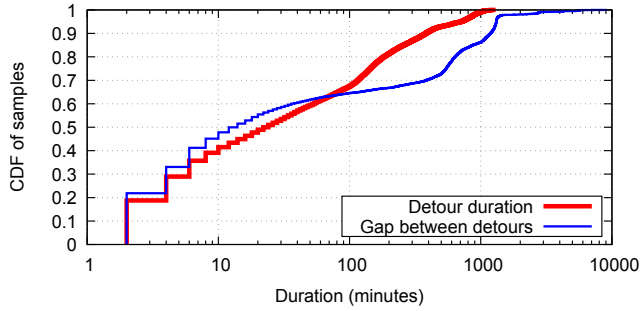

Figure 11: Distributions of Edge Fabric detour period lengths across (PoP, prefix) pairs and of time between detours.

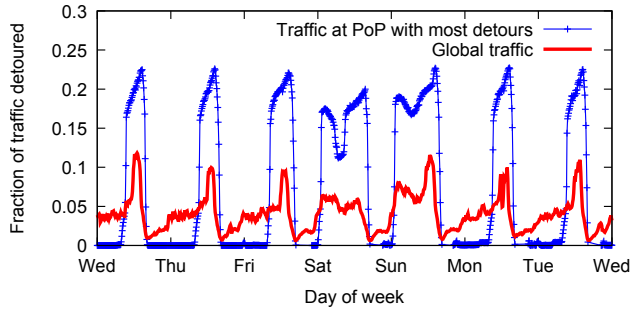

Figure 12: Fraction of traffic detoured by Edge Fabric across 20 PoPs and at the PoP with the largest fraction detoured.

from each interface to avoid overloading it. During our evaluation period, Edge Fabric detoured traffic from 18% of interfaces at least once, and it detoured 5% of interfaces for at least half the period. Figure 11 shows how long each period of detouring lasts, and how long the periods are between detours for a given (PoP, destination prefix). The median detour lasts 22 minutes, and 10% last at least 6 hours. Interestingly, the median time between detours is shorter–only 14 minutes–but the tail is longer, with a gap of more than 3 hours 36% of the time and a sizable fraction of gaps long enough to suggest detouring during a short daily peak. Figure 12 shows, over time, the fraction of traffic detoured across 20 PoPs and the fraction of traffic detoured at the PoP (in this set of 20) that detours the highest fraction of its traffic. The global and PoP detour volumes display diurnal patterns and remain a small fraction of overall traffic, leaving spare capacity to absorb detours, as PoPs always had at least 45% of their transit capacity free. Edge Fabric enables PoPs to dynamically detour traffic from interfaces that would otherwise become heavily overloaded (see Figure 6) by taking advantage of available capacity elsewhere.

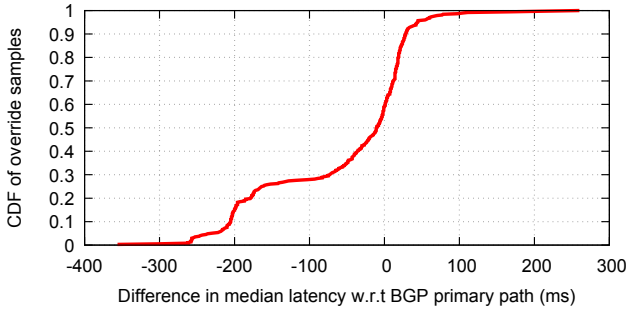

Figure 13: Performance difference between best paths chosen based on BGP policy and chosen based on alternate path measurements. Graph depicts latency-selected path minus policy-selected path, and so a negative value indicates that Edge Fabric's override achieved better performance (lower latency).

7.3 Evaluating Performance-Aware Routing

This section investigates how performance differs across paths to the same prefix and whether our system can use this information to select paths that outperform the ones BGP prefers. We use 7 days of measurements of production traffic from 4 PoPs where we deployed alternate path measurements (§7.1), including measurements of every prefix served from any of those PoPs, with a total of over 350M alternate path measurements to 20K ASes and an average of 8000 measurements per ⟨alternate path, destination prefix⟩.

What is the performance impact of using measurements of alternate paths to inform better decisions? We use the measurements to override BGP decisions for actual production user traffic at PoP-2. AltPath identified 400 destination prefixes for which an alternate route has a median latency at least 20ms faster (and loss was no worse) than BGP's preferred path (§7.1). Edge Fabric consumed this information and injected overrides at the PoP, steering these prefixes' production traffic (minus the small amount detoured for AltPath) to these routes. We left these overrides in place for 24 hours.

Figure 13 shows the achieved performance for prefixes redirected because of better median latency, comparing the performance of AltPath's designated path (now carrying the majority of traffic) versus the path preferred by our default BGP policy (now carrying a small amount of traffic for AltPath measurements). For these prefixes, 45% of prefixes achieved a median latency that was better by at least 20ms (even with most traffic shifted to the path, which potentially degrades performance), and 28% of prefixes improved by at least 100ms. On the other hand, some overrides did not work as expected. For example, 17% of prefixes experienced a median latency at least 20ms worse than the default path, and 1% were worse by at least 100ms. If the system dynamically adjusted routing (as opposed to our one-time static overrides), these results signal the potential for oscillations. The vast majority of these cases were prefixes moved from peer routes to transit provider routes.

When overrides result in worse performance, we speculate that this difference is likely a combination of two factors: (1) a path's performance is a function of the load placed onto it, and (2) a path's

performance can change over time. As future work, we intend to investigate these issues by conducting more granular shifts of traffic, so that we can understand how the load that we place onto particular paths impacts performance. In addition, this challenge points to the need for a robust and reactive control system that decides whether to shift traffic based not only on measurements for individual prefixes, but also on whether other traffic is being shifted to or from shared paths, historical information, and other factors that can impact the shift. Towards this goal, we plan to further increase the measurement sampling rate of AltPath, such that a controller could have sufficient samples to evaluate performance over a short timescale.

AltPath also identified 700 prefixes for which BGP's 3rd preference out-performed its 2nd preference, and so we configured Edge Fabric to use the 3rd preference when it needed to detour the prefixes to prevent congestion. However, during our experiment, fewer than 5% of these prefixes were detoured by Edge Fabric. We compared the performance of the selected detour path to AltPath measurements of BGP's 2nd preference (Edge Fabric's default detour) during these detours and found that all but 2 of the prefixes detoured achieved better performance. This result indicates that AltPath can help limit the performance hit experienced by prefixes that Edge Fabric detours due to congestion.

Does AltPath accurately capture end-to-end performance? We conducted a controlled experiment using the PEERING testbed [24]. The testbed peers directly with Facebook across the public AMS-IX fabric, and PEERING announcements also reach Facebook via PEERING's transit providers. We operated clients on IP addresses in a PEERING prefix that we announced both directly to Facebook and via transit. The clients continuously used HTTP to fetch 515KB and 30KB files from Facebook at a frequency that keeps the client's CPU and bandwidth utilization below 20%. Over time, we used Linux's traffic control framework to induce 40ms increased latency on traffic arriving directly from Facebook, via transit, or both. We used AltPath measurements to estimate the difference in performance (i.e., the difference in induced latency) on the direct and transit paths over 5-minute intervals. AltPath identified the difference in induced latency to within 2.2ms of the induced difference in all 5-minute intervals during an 18 hour experiment (except in the 5-minute intervals during which induced latency was in flux), with an average error of 0.6ms. This level of accuracy is sufficient to allow Edge Fabric to compare performance across paths when making detour decisions.

8 OPERATIONAL EXPERIENCE

Edge Fabric has evolved over years in response to growth at our PoPs and from realizations derived from operational experience. Our current design of Edge Fabric focuses on providing the flexibility that we require to handle different egress routing scenarios, but prefers well understood techniques and protocols over more complex approaches whenever possible.

8.1 Evolution of Edge Control

As the size and number of our PoPs have continued to grow, we strive for a simple, scalable design. These desired traits have required the continuous evaluation and improvement of different pieces of our design and the careful consideration of how a design decision will impact us in the long-term.

8.1.1 From stateful to stateless control. Our current implementation of Edge Fabric is stateless, meaning that it makes its allocation and override decisions from scratch in each 30 second cycle, without being aware of its previous detours. This approach has a number of advantages stemming from the simplicity of the design. For instance, because the stateless allocator begins each cycle by gathering all information it needs and projecting what utilization will be if the controller does not intervene (§5.1.2), it is straightforward to test, restart, or failover the controller. The controller only needs to calculate what traffic should be moved given its inputs and projection, and can be tested by simply providing input scenarios and checking its decision (§5.4).

In comparison, our previous stateful implementation required recording the allocator's state after each round both locally and remotely. If the controller was restarted due to an upgrade or a failure, it had to recover its previous decisions from a remote log, increasing complexity. In addition, the stateful controller's decision process was more complicated, as the controller not only had to decide which prefixes to shift when interfaces were overloaded but also which existing overrides to remove given current interface load. Because the stateful controller would not consider removing overrides until interface utilization dropped below a threshold, it could not backtrack while interface utilization was still increasing, and its options for further detouring were restricted by the impact of its previous actions. In some cases, the controller would shift a prefix to a detour interface, only to have the detour interface become overloaded in a subsequent cycle (due to the natural growth of traffic), requiring that the prefix be shifted yet again. Maintaining proper accounting of these states and decisions complicated the implementation and testing, since the logic and tests had to reason about and inject cascading decisions and states, ultimately providing the motivation for the stateless redesign.

8.1.2 From host-based to edge-based routing. Our current implementation of Edge Fabric uses BGP to enact overrides (§5.3) and only requires hosts to signal which flows require special treatment, such as those used for alternate path measurements (§§ 6.1 and 6.2). In comparison, previous implementations of Edge Fabric relied on host-based routing to enact overrides. In this model, the controller installed rules on every server in the PoP. These rules applied markings to traffic destined towards different prefix. Corresponding rules at PRs matched on these markings to determine which egress interface a packet should traverse, bypassing standard IP routing.

During the host-based routing era, Edge Fabric evolved through three different marking mechanisms in production: MPLS, DSCP, and GRE. (Our MPLS based implementation went a step further than what Edge Fabric does today by routing all egress traffic based on decisions made at the hosts, effectively shifting all IP routing decisions away from our PRs.) MPLS and DSCP were compatible with our early PoP architectures, in which we strived for balanced peer and transit connectivity across PRs, and any traffic that required detouring was sent to transit. Since traffic was subject to ECMP, all PRs had identical rules for the same peers (e.g., DSCP value *12* would detour traffic to transit X on all PRs). However, as

our PoP architectures grew, we increasingly had imbalanced transit and peering capacity across PRs and wanted control of which PR traffic egressed at, and so we switched to using GRE tunnels between servers and PRs.

From our experience with these mechanisms, we have found that it is non-trivial to obtain both host software and vendor software that provide fast and robust support for these tunneling protocols. Shifting the responsibility of *routing* traffic via a specific egress interface to end-hosts makes debugging and auditing the network's behavior more difficult, as configuration must be inspected at multiple layers. Further, when interfaces fail or routes are withdrawn, end-hosts must react quickly to avoid blackholing traffic, making synchronization among end-hosts, PRs, and controllers critical.

In comparison, Edge Fabric does not require hosts to be aware of network state and reduces synchronization complexities by injecting overrides to PRs at the edge of the network. In addition, this approach empowers PRs to invalidate controller overrides to prevent blackholing of traffic, since an interface failure will cause the PR to begin routing traffic to the next best route.

We believe our current edge-based approach provides us with many of the advantages of host-based routing with minimal complexity. While host-based routing gives hosts more control of how packets are routed, the additional flexibility is not currently worth the added complexity for the following reasons. First, our measurements demonstrate that the majority of traffic can use paths selected by BGP's standard process, with Edge Fabric overriding BGP only for a small portion of traffic in order to avoid causing congestion (§7.2) or to improve performance (§7.3). Second, our approach to overriding destination-based routing allows servers to tag select flows (§6.1) for special treatment, and it allows controllers to decide whether the routes for these flows should be overridden (§6.3). Although this decoupling limits the degree of control at servers, we believe that it results in a system that is simpler to design and troubleshoot, and that it provides sufficient flexibility for our intended use cases (§6.3). Third, because Facebook chooses to have traffic ingress, egress, and be served at the same PoP, the choices for routes are limited to decisions that can be signaled to and enacted at the PRs. If another PoP starts to provide better performance for a user network, the global traffic controller will redirect the traffic to that PoP.

8.1.3 From global to per-PoP egress options. Previously, Facebook propagated routes from external peers between PoPs in an iBGP mesh, such that a user's traffic could ingress via one PoP and egress via another. The ability to route traffic across the WAN to egress at a distant PoP can improve performance in some cases, but we had to design mechanisms to keep it from causing oscillations. For instance, some of the traffic on an overloaded egress interface may have ingressed at a remote PoP. If Edge Fabric overrode the route at the egress PoP to avoid congestion, the override update propagated to the ingress PoP. If we had allowed the override to cause the ingress PoP's BGP process to stop preferring the egress PoP, the projected demand for the overloaded egress interface could have dropped, which could cause Edge Fabric to remove the override, which would cause the ingress PoP to again prefer the original egress PoP, an oscillation. To prevent the override from causing the ingress PoP to change its preferred egress PoP, the controller set the BGP attributes of the override route to be equal to the original route. However, manipulating BGP attributes at the controller obfuscated route information, making it difficult to understand and debug egress routing. We disabled route redistribution between PoPs once we improved the accuracy and granularity of the global load balancer's mapping, and now traffic egresses at the same PoP as it ingresses. Since the global load balancer controls where traffic ingresses, it can spread traffic for a client network across PoPs that it considers to be equivalent to make use of egress capacity at multiple PoPs. This allows Facebook to avoid using backbone capacity to route traffic between PoPs and simplifies Edge Fabric's design.

8.1.4 From balanced to imbalanced capacity. As our PoPs have grown, we have had to extend our PoP design to handle corner-cases. As we increased the size and scale of our PoPs, we began to have more peers with imbalanced capacity (varying capacity to the same peer across PRs), partly due to the incremental growth of peering connectivity, but also because of inevitable failures of peering links and long recovery times. These imbalances created a problem because our ASWs use ECMP to distribute traffic evenly among PRs with the best paths. Instead of extending Edge Fabric to handle these imbalances, we could have chosen to use WCMP (Weighted Equal-Cost Multi-Path routing) at our ASWs and PR.

However, we chose to instead extend Edge Fabric to handle these capacity imbalances for a number of reasons. First, while a number of routing and switching chipsets support WCMP, it has far less adoption and support from our vendors than ECMP, making it riskier to adopt. Even with vanilla-ECMP, we have observed unequal traffic distributions (§5.4) and other erroneous or unexpected behavior. Second, since the WCMP implementations used by vendors are proprietary, we cannot predict how WCMP will behave, making projecting utilization and engineering traffic more difficult, and potentially increasing the complexity of Edge Fabric. Finally, WCMP implementations operate on the router's entire routing table, can take minutes to converge after an event (such as a link failure creating imbalanced capacity), and may not be efficient enough to balance traffic properly [33]. In comparison, Edge Fabric can identify the subset of prefixes with traffic and inject routes to mitigate the failure within seconds.

8.2 The Challenge of IXPs

Internet exchange points (IXPs) have been a focus in academia [2, 12], but they present challenges to a provider of Facebook's scale. In contrast to a dedicated private interconnect, a provider cannot know how much capacity is available at a peer's port, since other networks at the IXP may be sending to it as well. This limited visibility makes it harder to simultaneously avoid congestion and maximize interface utilization. Edge Fabric supports setting limits on the rate of traffic sent to a peer via a public exchange to avoid congesting a peer's public exchange connection (§3). Some exchanges report the total capacity of each peer's connection to the exchange fabric, but this information alone cannot be used to set a limit since we need to account for traffic that the peer will receive from other peers on the exchange. As a result, we set capacity constraints by contacting large public exchange peers and asking them for estimated limits on the maximum rate of traffic that we can send to them, as the peers have more insight into their interface

capacity and utilization. We handle overload on these connections using the same approach as a regular PNI interface, except we limit utilization per nexthop.

9 RELATED WORK

CDN traffic engineering. As a result of CDN traffic growth, researchers and engineers have proposed new traffic engineering solutions. A common approach in CDN traffic engineering, which Edge Fabric also employs, is centralized (SDN) control that combines network and traffic information to configure network devices.

Researchers have studied multiple ways to choose where a client's request should be directed (e.g., [7, 11, 32]), including using anycast [11] and DNS extensions [7] as mechanisms to direct clients over to "nearby" servers. These solutions target a different set of challenges than those described in §3 and are complementary to Edge Fabric. Facebook employs solutions to choose where a client should be directed, but still needs Edge Fabric to choose which route to use when sending data to the client.

More related to our work are Footprint [16], PECAN [28], Entact [32], and Espresso [31], which propose choosing which PoP and/or path a client should be directed to as a function of path performance. While Footprint focuses on shifting load of long-lived stateful client sessions between PoPs to avoid congestion and PECAN focuses on measuring performance and choosing ingress routes (from clients to servers), Edge Fabric was designed to shift client traffic between alternate egress routes (from servers to clients) to avoid congestion. The three proposals are complementary: our techniques could be applied to Footprint and PECAN, and vice-versa.

Entact and Espresso are most similar. Entact overrides BGP's default routing decisions through a well-designed approach that balances performance, load, and cost, evaluating the approach via emulation [32]. Similar to AltPath, Entact directs some traffic to alternate paths to measure their performance. We build on this idea, working through the details of deploying such a system in production, at scale, in a way that applies to all our services and users. For example, while Entact measured alternate path performance for individual IP addresses within prefixes, AltPath can assign at random or select flows across the address space, guarding against cases in which different addresses experience different performance and enabling future use of the same mechanism for application-specific routing. Entact uses active measurements (pings) to measure path performance, but is unable to find responsive addresses in many prefixes so can only make decisions for 26% of MSN traffic. The need for responsive addresses also limits the number of alternate paths that Entact can measure in parallel and keeps it from increasing the granularity of its decisions by deaggregating prefixes (both would require finding more responsive addresses). These atomic assignments may not be a good approximation of Entact's optimal traffic assignments, which assume a provider can split traffic to a prefix arbitrarily across multiple paths. By applying an approach similar to Entact's but based on passive measurement of production traffic, Edge Fabric uses Facebook's existing server-side measurement infrastructure to collect measurements that cover and are representative of our entire user base, can split prefixes to increase decision granularity, and can use as many paths in parallel as our peering routers can support. Finally, we expose challenges that arise in practice (§7.3), including the potential for oscillations, that do not occur in Entact's emulated evaluation.

Espresso is Google's SDN-based system to control egress routing [31]. Espresso and Edge Fabric are both designed by huge content providers needing to overcome challenges with BGP and BGP routers as they expand their PoP and peering footprint in the face of massive traffic growth. They take a similar top-level approach, centralizing control of routing while retaining BGP as the interface to peers. However, the two systems prioritize different tradeoffs in many other important design decisions, presenting an interesting case study of how the ranking of priorities can impact a design. Espresso uses a bespoke architecture to remove the need for BGP routers that support full Internet routing tables, whereas Edge Fabric relies on BGP and vendor BGP routers to build on existing experience and systems. Edge Fabric restricts the size of its multiple routing tables by isolating PoPs, such that the number of prefixes carrying user traffic per PoP is low (Figure 3). Whereas Facebook achieves simplicity by isolating prefix announcements, ingress, egress, and control to individual PoPs, Espresso uses a single global controller and can route traffic across the WAN to egress at distant PoPs, providing flexibility. Edge Fabric's controller pushes its egress decisions only to peering routers, allowing us to isolate Facebook's hosts from network state. Espresso, on the other hand, pushes routing decisions to hosts, maximizing flexibility but requiring a more sophisticated controller architecture and the continuous synchronization of routing state at hosts to prevent blackholing of traffic. Section 8.1 discusses these tradeoffs (relative to our goals) in more detail, based on our past experience with routing egress traffic between PoPs and with host-based routing (earlier Edge Fabric designs that were more similar to Espresso). Espresso includes approaches for mitigating some of the challenges that section describes. Our paper focuses on measurements and challenges of the BGP interconnectivity of a large content provider, and on the design and evaluation of a traffic engineering solution that integrates with existing peering routers.

B4 [14] and SWAN [13] centralize control of inter-datacenter networks to maximize utilization without hurting performance of high-priority traffic. Edge Fabric has a similar goal, and it also uses centralized control. However, the difference in the setting introduces new challenges. In particular, B4 and SWAN operate in a closed environment in which all hosts and network devices are under unified administration, and the majority of the traffic can tolerate delay and loss. In contrast, Edge Fabric controls egress traffic to networks and users outside its control. Further, much of the traffic is adaptive bitrate video, and it has soft latency demands far beyond the elastic traffic on inter-datacenter WANs. Fibbing centralizes control of legacy OSPF networks, injecting information to induce distributed routers to select desired intradomain routes [29]. Edge Fabric's controller similarly injects routes, but our interdomain setting provides much less visibility into or control over end-to-end paths. Edge Fabric could take advantage of richer mechanisms for traffic engineering, e.g., iSDX [12].

Performance monitoring. To inform CDN traffic engineering, existing techniques measure path performance by injecting measurement traffic into the network or by passively monitoring ongoing traffic. Active measurement techniques instrument clients

to collect measurements [4, 21] or dedicated measurement vantage points [1, 18, 19]. Passive measurement techniques can require (often expensive) monitoring functionality in network equipment [15, 17]. Edge Fabric conducts passive measurements at Facebook's servers, which allows (i) monitoring of all paths and services without the need to instrument millions of clients and (ii) the collection of richer metrics (e.g., SRTT) compared to on-path devices. Our finding that a small number of interfaces experience congestion is similar to previous IXP characterization studies [2, 8, 16].

10 CONCLUSION

Today's Internet traffic is dominated by a small number of big content providers. How they interact with other ASes largely shapes interdomain routing around the world.

This paper provides the first public details of the design, implementation, and operational experience of Edge Fabric, a system that steers vast amounts of content to the world. Edge Fabric augments BGP with measurement and control mechanisms to overcome BGP's lack of congestion- or performance-awareness. We designed it to be simple and scalable, taking advantage of centralized control, existing support in vendor software and hardware, and server-based measurements. Our results demonstrate that Edge Fabric successfully avoids congesting capacity-constrained interconnections, as well as show the potential for Edge Fabric's alternative path measurements to realize performance-aware interdomain routing.

BGP will be the Internet's interdomain routing standard for the foreseeable future. By sharing our 4 years of experience engineering our egress traffic, including a detailed look at opportunities and challenges presented by the Internet connectivity of today's large content providers, we hope that the limitations of BGP can be better understood and every Internet user's experience can be improved.

ACKNOWLEDGEMENTS

We thank Facebook colleagues for insights and feedback, including Steve Shaw, Manikandan Somasundaram, Lisa Guo, Alexander Kramarov, Bolek Kulbabinski, Vincent Mauge, Jimmy Ottosson, Callahan Warlick, Emre Cantimur, Yudong Yang, Martin Lau, Omar Baldonado, and, especially, Niky Riga and Varghese Vaidhyan. We thank our shepherd Olaf Maennel and the SIGCOMM reviewers. We thank the Espresso authors [31], especially K.K. Yap and Amin Vahdat, for informative exchanges to help highlight similarities/differences between our systems. Brandon Schlinker's research has been partially funded by the Facebook Graduate Fellowship. The PEERING testbed and Ethan Katz-Bassett's and Harsha Madhyastha's participation were funded in part by Facebook Faculty Awards and by the National Science Foundation (CNS-1406042, CNS-1351100, CNS-1413978, CNS-1563849, and CNS-1564242). Italo Cunha is funded in part by CNPq and FAPEMIG.

REFERENCES

[1] ThousandEyes: Network Intelligence Software. www.thousandeyes.com.
[2] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a Large European IXP. In *Proc. ACM SIGCOMM*, 2012.
[3] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, and R. Govindan. Mapping the Expansion of Google's Serving Infrastructure. In *Proc. ACM IMC*, 2013.
[4] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye. Analyzing the Performance of an Anycast CDN. In *Proc. ACM IMC*, 2015.
[5] Y.-C. Chiu, B. Schlinker, A. B. Radhakrishnan, E. Katz-Bassett, and R. Govindan. Are We One Hop Away from a Better Internet?. In *Proc. ACM IMC*, 2015.
[6] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the Impact of Video Quality on User Engagement. In *Proc. ACM SIGCOMM*, 2011.
[7] M. T. Fangfei Chen, Ramesh K. Sitaraman. End-User Mapping: Next Generation Request Routing for Content Delivery. In *Proc. ACM SIGCOMM*, 2015.
[8] N. Feamster. 2016. Revealing Utilization at Internet Interconnection Points. *CoRR* abs/1603.03656 (2016).
[9] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan. Reducing Web Latency: The Virtue of Gentle Aggression. In *Proc. ACM SIGCOMM*, 2013.
[10] T. Flach, P. Papageorge, A. Terzis, L. D. Pedrosa, Y. Cheng, T. Karim, E. Katz-Bassett, and R. Govindan. An Internet-Wide Analysis of Traffic Policing. In *Proc. ACM SIGCOMM*, 2016.
[11] A. Flavel, P. Mani, D. Maltz, N. Holt, J. Liu, Y. Chen, and O. Surmachev. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *Proc. USENIX NSDI*, 2015.
[12] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever. An Industrial-scale Software Defined Internet Exchange Point. In *Proc. USENIX NSDI*, 2016.
[13] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving High Utilization with Software-driven WAN. In *Proc. ACM SIGCOMM*, 2013.
[14] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a Globally-deployed Software Defined Wan. In *Proc. ACM SIGCOMM*, 2013.
[15] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet Inter-domain Traffic. In *Proc. ACM SIGCOMM*, 2010.
[16] H. H. Liu, R. Viswanathan, M. Calder, A. Akella, R. Mahajan, J. Padhye, and M. Zhang. Efficiently Delivering Online Services over Integrated Infrastructure. In *Proc. USENIX NSDI*, 2016.
[17] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman. One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon. In *Proc. ACM SIGCOMM*, 2016.
[18] M. Luckie, B. Huffaker, K. Claffy, A. Dhamdhere, and V. Giotsas. AS Relationships, Customer Cones, and Validation. In *Proc. ACM IMC*, 2013.
[19] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: an Information Plane for Distributed Services. In *Proc. USENIX OSDI*, 2006.
[20] S. Meinders. In *RIPE NCC Regional Meeting: Eurasia Network Operators Group (ENOG 11)*, 2016.
[21] A. Nikravesh, H. Yao, S. Xu, D. Choffnes, and Z. M. Mao. Mobilyzer: An Open Platform for Controllable Mobile Network Measurements. In *Proc. ACM MobiSys*, 2015.
[22] R. Sambasivan, D. Tran-Lam, A. Akella, and P. Steenkiste. Bootstrapping Evolvability for Inter-domain Routing with D-BGP. In *Proc. ACM SIGCOMM*, 2017.
[23] Sandvine. Global Internet Phenomena Report 2H2016. Available at: http://www.sandvine.com/trends/global-internet-phenomena.
[24] B. Schlinker, K. Zarifis, I. Cunha, N. Feamster, and E. Katz-Bassett. PEERING: An AS for Us. In *Proc. ACM HotNets*, 2014.
[25] J. Scudder, R. Fernando, and S. Stuart. RFC 7854: BGP Monitoring Protocol (BMP). http://www.ietf.org/rfc/rfc7854.txt.
[26] D. Sommermann and A. Frindell. Introducing Proxygen, Facebook's C++ HTTP framework. https://code.facebook.com/posts/1503205539947302.
[27] Y.-W. E. Sung, X. Tie, S. H. Wong, and H. Zeng. Robotron: Top-down Network Management at Facebook Scale. In *Proc. ACM SIGCOMM*, 2016.
[28] V. Valancius, B. Ravi, N. Feamster, and A. C. Snoeren. Quantifying the Benefits of Joint Content and Network Routing. In *Proc. ACM SIGMETRICS*, 2013.
[29] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford. Central Control Over Distributed Routing. In *Proc. ACM SIGCOMM*, 2015.
[30] D. Wing and A. Yourtchenko. RFC 6555 Happy Eyeballs: Success with Dual-Stack Hosts. http://www.ietf.org/rfc/rfc6555.txt.
[31] K. K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, V. Lin, C. Rice, B. Rogan, A. Singh, B. Tanaka, M. Verma, P. Sood, M. Tariq, M. Tierney, D. Trumic, V. Valancius, C. Ying, M. Kallahalla, B. Koley, and A. Vahdat. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *Proc. ACM SIGCOMM*, 2017.
[32] Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian. Optimizing Cost and Performance in Online Service Provider Networks. In *Proc. USENIX NSDI*, 2010.
[33] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat. WCMP: Weighted Cost Multipathing for Improved Fairness in Data Centers. In *Proc. ACM EuroSys*, 2014.