SCULPTOR Has Your Back(up Path): Carving Interdomain Routes to Services

Paper #156, 12 pages body, **TBD: 19** pages total

Abstract

Large cloud/content (service) providers serve an expanding suite of applications that are increasingly integrated with our lives, but have to contend with a dynamic public Internet to route user traffic. To enhance reliability to dynamic events such as failure and DDoS attacks, Service Providers overprovision to accommodate peak loads and activate emergency systems for shifting excess traffic. We take a different approach with SCULPTOR, which proactively advertises many prefixes for users so that Service Providers can use their global resources to meet generic objectives. SCULPTOR models Internet routing to solve a large integer optimization problem at scale using gradient descent. We prototyped SCULPTOR on a global public cloud and tested it in real Internet conditions, demonstrating that SCULPTOR handles dynamic loads 28% larger than other solutions using existing Service Provider infrastructure, reduces overloading on links during site failures by up to 40%, and enables Service Providers to route high-priority traffic with 2× less overloading.

1 Introduction

Cloud/content providers (hereafter Service Providers) enable diverse Internet applications used daily by billions of users. Traditionally, Service Providers used DNS and static BGP advertisements to define a single path from a user to a service, leaving the specific path largely up to the Internet to determine.

Increasingly, however, these services have diverse requirements that can be challenging to meet with a single path that Service Providers have little control over. For example, enterprise services have tight reliability requirements [55, 52], and new applications such as virtual reality require ≤ 10 ms round trip latency [76] and ≤ 3 ms jitter [102]. Complicating matters, Service Providers must meet these requirements subject to changing conditions such as peering link/site failures [30, 71], DDoS attacks [79, 111, 90], flash crowds [47, 60, 63], new applications/business priorities (LLMs), and route changes [77, 69, 36, 70, 84, 71]. Critically, such changes can cause *overload* if the Service Provider cannot handle new traffic volumes induced by the change, and DNS/BGP are slow mechanisms with which to change how traffic flows over paths. This overload can lead to degraded service for users, hurting reliability [78, 88, 56, 52], and may require manual intervention to remedy.

To try to meet different goals and respond to changing conditions, Service Providers adopt solutions that are either (a) too conservative, (b) too reactive, or (c) too specific. For example, Service Providers proactively overprovision resources [103, 1, 65], but we demonstrate using traces that bursty traffic patterns can require excessively conservative overprovisioning rates as high as 70% despite sufficient global capacity (§2.3). TIPSY responds to changing loads to retain reliability [71], but only does so reactively, which could lead to short-term degraded performance. AnyOpt and PAINTER proactively set up routes to optimize specific objectives such as steady-state latency [115, 55], but it is challenging to extend those approaches to other objectives (§5.4). It is also unclear both how to combine these approaches (*e.g.,* retaining low latency under changing traffic loads) and how to apply approaches from one domain to another (*e.g.,* applying egress traffic cost reduction systems [99] to ingress traffic).

We present Service Providers with a flexible framework, SCULPTOR (**S**couring **C**onfigurations for **U**tilization-**L**oss-and **P**erformance-aware **T**raffic **O**ptimization & **R**outing), which accepts as input cost, performance, and reliability objectives and outputs BGP advertisements and traffic allocations that help achieve those desired objectives. SCULPTOR is the first system that optimizes ingress interdomain routing objectives such as maximum link utilization, transit cost, and latency for *interdomain* traffic (existing systems optimize for intradomain traffic, *e.g.,* [48, 37, 72, 15]). We leave a formal characterization of the objectives, constraints, and problem sizes SCULPTOR covers to future work, and instead demonstrate its practical utility (§5). SCULPTOR computes BGP advertisements proactively, only placing live traffic on them after convergence, and hence does not run the risk of outages.

To solve each optimization problem, SCULPTOR efficiently searches over the large BGP advertisement search space (> $2^{10,000}$ possibilities) by modeling how different strategies perform, without having to predict the vast majority of actual paths taken under different configurations since predicting interdomain paths is hard and measuring them is slow (§3.3). SCULPTOR then optimizes these (modeled) performance metrics using gradient descent, which is appropriate in our setting due to the high dimension of the problem and the parallelism that gradient descent admits (§3.4). This modeling enables SCULPTOR to assess > 20M configurations (10,000× more than other solutions [115, 55]) while only measuring tens in the Internet (§5.4).

We prototype and evaluate our framework at Internet scale using the PEERING testbed [93] (§4), which is now deployed at 32 Vultr cloud locations [108]. Vultr is a global public cloud that allows our prototype to issue BGP advertisements via more than 10,000 peerings. We evaluate our framework on two specific objectives (computed separately): (a) optimizing latency under unseen traffic conditions, and (b) routing different traffic classes. We compare SCULPTOR's performance on both problems to that of an unreasonably expensive "optimal" solution (computing the actual optimal is infeasible).

For the first objective, we found that, compared to other approaches, SCULPTOR increases the amount of traffic within 10 ms of the optimal by 19.3% in steady-state (meeting that target for 95% of traffic) (§5.2.1), by 11% during link failure, and by 17% during site failure. SCULPTOR also reduces overloading on links during site failures by up to 41% over PAINTER, giving Service Providers more confidence that services will still be available during partial failure (§5.2.2). We also find that, by load balancing traffic on backup paths during peak times, we can satisfy high peak demands with the same infrastructure. SCULPTOR can handle flash crowds (*e.g.,* DDoS attacks) at more than 3× expected traffic volume, reducing the amount of overprovisioning that Service Providers need, thus reducing costs.

For the second objective, SCULPTOR routes bulk low-priority traffic in ways that avoid congesting high priority traffic, achieving near-optimal latency and reducing congestion by up to 2× (§5.3).

After decades of intradomain traffic optimization using programmable networking primitives such as virtual output queueing for differentiated service [82] (and others, §7), SCULPTOR brings such control closer to realization in the interdomain setting with no buy-in from other networks. Service Providers can use flexible frameworks such as SCULPTOR to help bring us the resilient, performant service that our diverse applications increasingly need.

2 Motivation

2.1 Variable, Evolving Goals

Evolving Internet use cases are pushing Service Providers to meet diverse requirements for their applications. For example, Service Providers increasingly host mission-critical services such as enterprise solutions [55], which require high reliability and are predicted to be a $60B industry by 2027 [46]. Gaming is a similarly important industry generating more revenue than the music and movie industries combined [5], but instead requires latency within 50 ms [76]. Service Providers are also expanding the set of services they provide—for example, CDNs that traditionally hosted static content are pivoting to offering services like compute [19, 28, 27].

Moreover, as use cases evolve, Service Providers increasingly need to meet performance requirements for *ingress* traffic since that traffic includes, for example, player movements in real-time games, voice and video in enterprise conferences, and video/image uploads for AI processing in the cloud [17, 74]. This reality is a departure from traditional CDN traffic patterns where ingress traffic was primarily small requests and TCP acknowledgments.

Despite Service Providers' efforts to meet variable service requirements [71, 51, 35, 58, 75, 55, 65], meeting requirements is still challenging due to issues *outside* Service Provider control.

First, Service Providers lack control over which ingress path traffic takes since BGP, the Internet's interdomain routing protocol, computes paths in a distributed fashion, giving each intermediate network a say in which paths are chosen and which are communicated to other networks. BGP only chooses a single path, and that path may not be the best one for meeting a given requirement. Service Providers cannot even unilaterally control the part of the path closest to them—upgrading peering capacity requires coordination among multiple parties [95].

Second, dynamic factors outside Service Provider control make satisfying requirements even harder. Peering disputes can lead to congestion on interdomain links and inflated paths [30, 21, 25], and DDoS attacks still bring down sites/services [90, 78, 88, 56], despite the considerable effort in mitigating DDoS attack effects [79, 111]. Moreover, recent work [63, 84, 15] and blog posts [47, 60, 77, 69, 36, 70] show that user traffic demands are highly variable due to flash crowds and path changes and so can be hard to plan for.

2.2 Routing Traffic to Service Providers

One way of meeting service requirements, despite this lack of control, is to (a) configure a good set of interdomain paths and (b) balance traffic on these paths to satisfy requirements. Existing systems balance traffic over egress interdomain paths [95, 114, 37, 59], but it remains unclear how to configure and

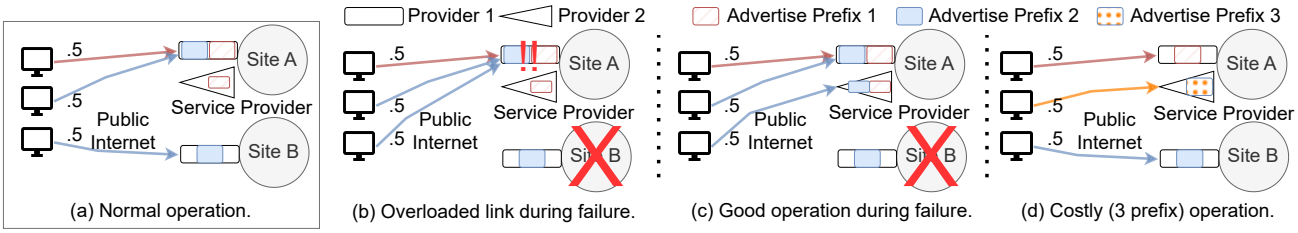

Figure 1: In normal operation traffic is split between two sites by directing half the traffic to each prefix (a). Even though there is enough global capacity to serve all traffic when site B fails, there is no way to split traffic across multiple providers given available paths, leading to overload (b). A proactive solution is to advertise prefix 2 to an additional provider at site A, allowing traffic splitting across the two links (c). A simpler but prohibitively expensive solution is to advertise one prefix per peering (d).

balance traffic across a good set of *ingress* interdomain paths.

Today, most Service Providers either use `anycast` prefix advertisements to provide relatively low latency and high availability at the expense of some control [8, 54, 119, 118], or `unicast` prefix advertisements to direct users to specific sites [98, 57, 95, 13].

`Anycast`, where Service Providers advertise a single prefix to all peers/providers at all sites, offers some natural availability following failures since BGP automatically reroutes most traffic to avoid the failure after tens of seconds [119]. Prior work shows that this availability comes with higher latency in some cases [64, 54]. `Unicast` gives clients lower latency than `anycast` by advertising a unique prefix at each site [98, 13], but can suffer from reliability concerns, potentially taking as much as an hour to shift traffic away from bad routes [55].

2.3 Limitations of Current Approaches

Too Specific: Recent work has similarly noted that, by offering better interdomain routes, Service Providers can achieve better performance [30, 8, 115, 118, 55]. These solutions advertise different prefixes to subsets of all peers/providers to offer users more paths. However, these solutions optimize specific objectives, mostly steady-state latency, and it is unclear how to extend their approaches to other objectives (§5.4), especially those with capacity constraints.

Figure 1 shows how this specific focus on a single objective could lead to reliability issues, *e.g.,* during a site failure. In normal operation, user traffic is split evenly across two prefixes and achieves low latency (Fig. 1a). However when site B fails, BGP chooses the route through Provider 1 for all traffic, causing link overutilization and subsequent poor performance for all users (Fig. 1b). In Section 5 we show that this overload happens in practice for state-of-the-art systems that provide low (steady-state) latency from users to Service Provider networks such as `AnyOpt` [115] and `PAINTER` [55].

Too Reactive: To enhance reliability, some Service Providers propose systems that react to changing conditions—for example, `TIPSY` drains traffic from overloaded links/sites [71]. However, reactivity still requires time to change to a new state, which can lead to short-term service degradation and additional uncertainty at a time where unexpected things are already occurring. A more desirable approach would be to never change announcements carrying live traffic, which is known to sometimes lead to outages [6, 49, 10, 65].

Too Conservative: Another approach is to overprovision resources to handle transient peak loads [103, 1, 65]. Figure 2 demonstrates however, using longitudinal link utilization data from OVH cloud [85], that doing so can incur excessive costs. Meta also stated that solely installing extra capacity without shifting traffic is not always a feasible solution since peering capacity cannot be unilaterally upgraded [95], and Microsoft said the same because of lead time [30].

Upgrading capacity has fixed costs (*e.g.,* fiber, router backplane bandwidth, line card upgrades, CDN servers near peering routers) and variable costs (*e.g.,* power, transit), both of which scale with demand and Service Providers strive to keep low [86, 1, 15, 99]. We model deployment cost as correlated with the peering capacity over all links/sites, even though the actual relationship may be complex and additional peering capacity itself is not prohibitively expensive.

To generate Figure 2 we first split the dataset into successive, non-overlapping 120-day planning periods and compute the 95^{th} percentile ingress link utilization ("near-peak load") for each link and for each period. The dataset reports utilizations over approximately 5-minute intervals. We then simulate assigning future capacities from period to period by setting each link's capacity for the next period as the near-peak load in the current period multiplied by some overprovisioning factor (varying the factor on the X axis). We then compute the average link utilizations in the next period and the total number of links on which we see $\geq$ 100% utilization in at least one 5-minute interval.

Figure 2 plots the median utilization across links and periods and the number of congestive events as we vary the overprovisioning factor. Figure 2 shows that overprovisioning to accommodate peak loads introduces a tradeoff between inefficient utilization and overloading. Low overprovisioning factors between 10% and 30% lead to more efficient utilization (60%-70%) but lead to thousands of congestive events. High overprovisioning factors lead to far less overloading, but only roughly 50% utilization.

Efficient, Proactive Planning is Possible: Despite these limitations, Figure 1c shows that by advertising prefix 2 to

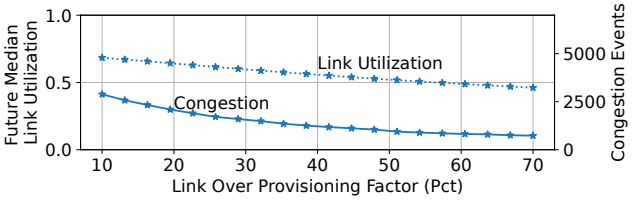

Figure 2: Planning for peak loads to avoid overloading requires inefficient overprovisioning.

provider 2 at site A a priori, the Service Provider can split traffic between the two links during failure, avoiding overloading without (a) reacting to the failure and (b) upgrading deployment capacity.

2.4 Key Challenges

Since a Service Provider has lots of global capacity, dynamically placing traffic on paths to optimize performance objectives subject to capacity constraints would therefore be simple if all the paths to the Service Provider were always available to all users as in Figure 1d. However, making paths available uses IPv4 prefixes, which are monetarily expensive and pollute BGP routing tables. The solution in Figure 1d uses 50% more prefixes than the solution in Figure 1c and, at $20k per prefix for 10,000 peerings would cost $200M [80]. IPv6 is not a good alternative, as IPv6 routing entries take 8× the amount of memory to store in a router so would pollute global routing tables even more. Prior work noted the same but found that advertising around 50 prefixes was acceptable [115, 55], and most Service Providers advertise fewer than 50 today according to RIPE RIS BGP data [101]. We also verified with engineers at six Service Providers that this challenge important to address.

Since we cannot expose all the paths by advertising a unique prefix to each connected network, we must find some subset of paths to expose.

Finding that right subset of paths to expose that satisfies performance objectives, however, is hard since there are exponentially many subsets to consider, and each subset currently needs to be tested (*i.e.,* advertised via BGP) to see how it performs. The number of subsets is on the order of 2 to the number of ingresses Service Providers have, which, for some Service Providers (including Vultr, which we measure) [105], is $> 2^{10{,}000}$. As measuring this many advertisements is intractable, we have to predict how different subsets of paths perform, which is challenging since interdomain routing is difficult to model [96, 71].

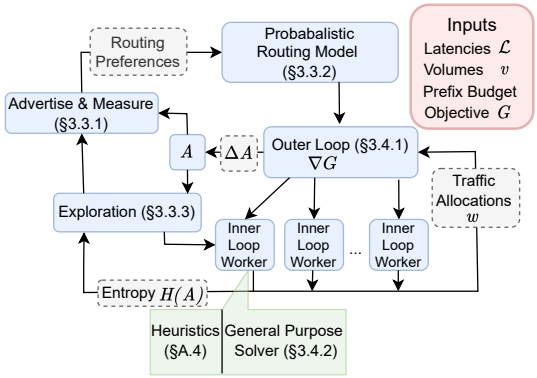

Figure 3: SCULPTOR advertisement computation overview.

3 Methodology

3.1 SCULPTOR Overview

SCULPTOR's goal is to find an advertisement strategy that gives users good interdomain paths (relative to some objective) and a traffic allocation to those paths. SCULPTOR computes BGP advertisements proactively, only placing live traffic on them after convergence. For objectives we consider, resulting optimization problems are often large with more than 100M constraints and 2M decision variables (§3.2.2). Our framework breaks this challenging problem into manageable components — Figure 3 shows the interactions among these components.

Minimizing an objective function requires evaluating it with many different inputs, but performing such measurements (*i.e.,* advertising prefixes) could take years at our problem size (§3.3.1) and so is not scalable. Instead, we estimate the objective using a "Probabilistic Routing Model" (§3.3.2) and update this model over time by "Advertising & Measuring" a small number of advertisements in the Internet, partially using entropy-based "Exploration" (§3.3.3).

We then minimize the objective function using gradient descent ("∇G") which, at each iteration, requires solving millions of sub-problems for traffic allocations ("Inner Loop Workers", §3.4.1). These sub-problems can either be solved exactly with a "General Purpose Solver" (§3.4.2) or with efficient heuristics (Appx. A.4).

3.2 Problem Setup and Definitions

3.2.1 Setting

Service Providers offer their services from tens to hundreds of geo-distributed sites [73]. The sites for a particular service can serve any user, but users benefit from reaching a low-latency site for performance. Sites consist of sets of servers that have an aggregate capacity. Sites can forward requests to other sites via private WANs as in FastRoute, but such forwarding is undesirable since it uses valuable private WAN capacity [52].

Service Providers also connect to other networks at sites via dedicated links or shared IXP fabrics which we call peering links. Each such link also has a capacity. When utilization of a site or link nears/exceeds the capacity, performance suffers, so Service Providers strive to avoid high utilization [30, 15]. Resources can also fail completely due to, for example, physical failure and misconfiguration.

Users route to the deployment through the public Internet to a prefix over one of the peering links via which that prefix is advertised. The path to a prefix (and therefore peering link) is chosen via BGP. We fix the maximum number of allowable prefixes according to the Service Provider's budget, which is generally much less than the number of peerings.

We model interdomain paths from users to the Service Provider as non-overlapping, except at the peering link through which each path ingresses to the Service Providers' deployment. We assume the peering link is the bottleneck of each path for two reasons. First, this link increasingly represents the most important part of the path for Service Providers due to Internet flattening [4]. Second, other parts of the path are less interesting from an optimization perspective—last-mile bottlenecks will be common to all paths for a user, and existing systems optimize intradomain paths in other networks. Handling unexpected bottlenecks can also be viewed as an "Unseen Scenario" (§3.2.3) so SCULPTOR may gracefully handle violations of this assumption.

A user may have many paths to a site through different links, each with different capacities and latencies. A path to a given prefix will be through one of the corresponding ingress links over which that prefix is advertised.

We consider users at the granularity of user groups (UG*s*), where a UG refers to user networks that route to the Service Provider similarly, but could mean different things in different instantiations of our system (*e.g.,*/24 IPv4 prefixes, metros). UG*s* generate steady-state traffic volumes, $v_{\texttt{UG}}$, and the Service Provider provisions capacity at links/sites to accommodate this load. A system run by the Service Provider measures latency from UG*s* to Service Provider peering links l, which is a reasonable assumption [9, 13, 24, 34, 115, 55].

We assume that the Service Provider has some technology for directing traffic towards prefixes that SCULPTOR can use to direct a UG. Examples include DNS [98, 13, 8, 57], multipath transport [89, 22, 23], or control-points at/near user networks [92, 55]. DNS offers slow redirection due to caching [55] but is the most readily deployable by the largest number of Service Providers, whereas Service Provider-controlled appliances offer precise control but may not be a feasible option for some Service Providers. Service Providers with stronger incentives to provide the best service to users will invest in better options with more control, and eventually multipath transport will see wide enough deployment to be used by all Service Providers. Today, MPTCP is enabled by default in iOS [3] and Ubuntu 22 [31], and MPQUIC can be used in any application [117].

3.2.2 General Formulation

The problem of finding advertisement configurations that admit good assignments of user traffic to paths is a multivariate optimization over both configurations and traffic assignments.

We represent an advertisement configuration A as a binary vector. Each entry indicates whether we advertise/do not advertise a particular prefix to a particular peer/provider (similar to prior work [115, 55]). Implementing this advertisement configuration (*i.e.,* advertising prefixes via BGP sessions) results in routes from users to the Service Provider. The assignment of traffic to resulting routes from this advertisement configuration is given by the nonnegative real-valued vector w, whose entries specify traffic allocation for each user along each route.

Announcing a configuration will result in some set of routes (although knowing exactly which routes a priori is challenging), and these routes define which ingress link a UG uses for a prefix. We can think of this process as a routing function R that takes an advertisement configuration A and outputs a map from $\langle \texttt{prefix, UG} \rangle$ pairs to ingress links. For example, say $R(A) = f_A$, and $f_A(p, \texttt{UG}) = l$ — this notation means that the output of an advertisement configuration A under routing R is a function f_A that tells us that users UG reach prefix p via link l. It could be that a configuration leads to no route for some UG to some prefix. We define a function e such that $e(R(A)(p, \texttt{UG})) = 1$ when there is some route for UG to prefix p under configuration A, and 0 otherwise.

Now suppose the overall metric we want to minimize is G which is a function of both configurations and traffic assignments. Examples include traffic cost, average latency, maximum latency, and their combinations (see Appx. A.3 for a discussion of which metrics may work better than others). The joint minimization over configurations and traffic assignments can then be expressed as the following.

$$\begin{aligned} \min_{A,w} \quad & G(R(A), w) \\ \text{s.t.} \quad & w(p, \texttt{UG}) \geq 0; \quad A(p,l) \in \{0,1\} \quad \forall \texttt{UG}, p, l \\ & \sum_p w(p, \texttt{UG}) e(R(A)(p, \texttt{UG})) = v(\texttt{UG}) \quad \forall \texttt{UG} \end{aligned} \tag{1}$$

The first constraint requires that traffic assignments be nonnegative and that configurations are binary. The second constraint requires that all user traffic $v(\texttt{UG})$ is assigned, and none is assigned to nonexistent paths. Capacity constraints depend on the choice of G.

3.2.3 Specific Objectives

In our evaluations we focus on two specific objectives: (1) minimizing latency and maximum link utilization in unseen scenarios, and (2) optimally routing different traffic classes. We believe the framework will accommodate objectives that have been met in other networking settings