SCULPTOR Has Your Back(up Path): Resilient, Efficient Interdomain Routing for Large Service Providers

Paper #559, 12 pages body, 19 pages total

Abstract

Large cloud and content (service) providers help 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, large service providers overprovision to accommodate peak loads and reactively activate emergency systems for shifting excess traffic. We take a different approach with `SCULPTOR`, which proactively installs routes so that Service Providers can use their global resources to handle dynamic scenarios without excessive overprovisioning. `SCULPTOR` ensures users have many interdomain routes to the Service Provider, and moves excess traffic to backup paths to avoid overloading and minimize latency. `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 17% on average, and enables service providers to achieve low latency objectives for up to 17% more traffic.

1 Introduction

Compiled on 2024/08/19at 11:52:37

Cloud and content providers (hereafter Service Providers) enable diverse Internet applications used daily by billions of users. Diverse applications have diverse requirements and exist in different ecosystems creating unique performance, reliability, and cost surfaces over which Service Providers must optimize. For example, enterprise services have tight reliability requirements [**? ?**], whereas new applications such as virtual reality require ≤ 10 ms round trip latency [**?**] and ≤ 3 ms jitter [**?**]. It may therefore make sense for different Service Providers to allocate resources in different ways to meet these requirements to varying degrees.

A variable Service Providers can optimize to meet these goals is the end-to-end path between user and Service Provider networks. For example, Service Providers shift changing egress demands across links to meet performance objectives [**? ? ? ? ?**]. However, it is difficult to similarly shift changing ingress demands since ingress traffic can only be controlled indirectly, by directing users to different prefixes or by changing announcements

Complicating matters, Service Providers must meet these requirements in a dynamic Internet. There are many changing conditions outside of Service Providers' control that Service Providers have to prepare for such as peering link/site failures [**? ?**], DDoS attacks [**? ? ?**], flash crowds [**? ? ?**], new applications/business priorities (LLMs), and route changes [**? ? ? ? ? ?**]. Critically, such changes can cause *overload* if the service cannot handle new traffic volumes induced by the change. This overload can lead to degraded service for users [**? ? ? ?**].

In response to these varied goals and changing landscapes, Service Providers generally propose solutions that are either (a) too conservative, leaving optimization potential on the table, (b) too specific, requiring a bespoke solution for each Service Provider/use case, or (c) too reactive, leading to performance reduction in the short to medium term. For example, Service Providers proactively overprovision resources [**? ? ?**], but we demonstrate using traces that bursty traffic patterns can require conservative overprovisioning rates as high as 70% (**??**). Systems such as `AnyOpt` and `PAINTER` try to optimize specific objectives such as steady-state latency [**? ?**], whereas `FastRoute` and `TIPSY` reactively try to retain reliability in the face of changing loads [**? ?**]. It is unclear both how to combine these approaches into one system, and how to apply approaches from one domain to another — for example: applying egress traffic cost reduction systems to ingress traffic [**?**].

\iffalse

Cloud and content providers (hereafter Service Providers) enable Internet applications used daily by billions of users. The applications have increasingly stringent performance and