Quantifying the Benefits of Joint Content and Network Routing

Vytautas Valancius, Bharath Ravi, Nick Feamster, and Alex C. Snoeren†

Georgia Tech and †UC San Diego

ABSTRACT

Online service providers aim to provide good performance for an increasingly diverse set of applications and services. One of the most effective ways to improve service performance is to replicate the service closer to the end users. Replication alone, however, has its limits: while operators can replicate static content, wide-scale replication of dynamic content is not always feasible or cost effective. To improve the latency of such services many operators turn to Internet traffic engineering. In this paper, we study the benefits of performing replica-to-end-user mappings in conjunction with active traffic engineering. We present the design of PECAN, a system that controls both the selection of replicas (“content routing”) and the routes between the clients and their associated replicas (“network routing”). We emulate a replicated service that can perform both network and content routing by deploying PECAN on a distributed testbed. In our testbed, we see that jointly performing content and network routing can reduce round-trip latency by 4.3% on average over performing content routing alone (hence, reducing service response times by at least tens of milliseconds) and that most of these gains can be realized with no more than five alternate routes at each replica.

1. Introduction

Online service providers (OSPs) such as Facebook and Google are offering an increasingly diverse set of interactive online services ranging from social networking services to online productivity tools. Consumers expect these services to be responsive, and OSPs are continually implementing optimizations to improve their performance with significant impact to their bottom lines. Google research showed that a 500-millisecond increase in latency caused a 20% traffic drop [1], while it was reported that a latency increase of 100 milliseconds can produce a 1% drop in revenue for Amazon [27]. Accordingly, recent years have seen many optimizations to accelerate the delivery of such services, ranging from better transport protocols [37] to browser enhancements [2, 24] to new compression and site optimization algorithms [4, 14].

One of the most common and effective ways to improve the performance of an online service is to decrease the path latency between the service and its clients by replicating the service at many geographic locations. Operators of replicated online services continually map clients to the data center that offers the best end-to-end service performance; this process is called content routing. Thanks to content distribution networks (e.g., Akamai and Limelight), replication and content routing are prevalent for static content, but replicating full-featured Web service logic that serves dynamic content can be difficult and costly, and, as a result, large-scale replication is not always feasible. Indeed, even the most popular Web service providers often host back-end logic at only a handful of sites. For example, Facebook serves all dynamic content from only four data centers, and Amazon serves its EC2-based properties from seven.

On the other hand, past work on detour routing [35], overlay routing [10], and multi-homing [7–9] suggests that the default wide-area Internet paths between a client and any given service replica may be suboptimal: Network operators may be able to optimize the network routing at each replica site to improve performance. In practice, however, network operators have little visibility into the performance that a given replica would offer to a particular client or set of clients. The operations teams that perform network and content routing are often distinct, and frequently do not coordinate with one another [25]. The operators of major OSPs that we surveyed stated that their service replica operators and network operators have only limited cooperation, and they do not attempt to reap the benefits of jointly optimizing content and network routing. Client performance suffers from this lack of coordination: operators of service replicas currently have no visibility into the performance or cost of alternate network paths between a service replica and its clients, so they optimize replica mapping based upon the current network paths that have been exposed as a result of network operators’ traffic engineering optimizations. On the other hand, network operators, who do have access to alternate wide-area paths, have little insight into the application-level performance these alternate paths might provide.

To bridge this divide, we design and evaluate PECAN (Performance Enhancements with Content And Network routing), a system that performs joint content and network routing for dynamic online services. PECAN enables joint content and network routing for online services by augmenting an OSP’s existing content routing framework to provide a diverse set of wide-area routes between each interactive service replica and its clients. To ensure that PECAN does not harm the performance of any existing service, it explores alternate wide-area routes using separate IP prefixes; clients can always reach the online service either via the default wide-area Internet routes or via PECAN’s routes.
We measured the performance benefits that PECAN can achieve in practice. We emulate an online service provider’s infrastructure by placing replicas at the Transit Portal (TP) locations [5] and clients at nodes in the PlanetLab testbed. TP allows us to emulate an OSP with a five geographically diverse, U.S.-based points-of-presence (PoPs), each of which provides access to many alternate wide-area paths to clients. There are many ways to measure performance; one metric is Web page load time, but accurately measuring page load time is challenging, as it requires instrumenting each client with browser software—a difficult task for a large-scale measurement study. Instead, we focus on network latency (i.e., round trip time), because many online service providers have identified latency as a key factor governing a user’s experience [15, 16].

Using three months of data from our testbed, we find that, when compared to performing content routing alone, using joint routing improves improve performance for about 35% of clients. Moreover, over 20% directly benefit from joint content and network routing, as they achieve better performance by employing an alternate route to a different replica than they would have selected if only default routes were available. In our experiments, we find that applying content routing alone decreases service latency by 16.75% on average relative to an optimally placed non-replicated service. Joint routing delivers an additional 4.3% (or about 5 ms) average round-trip latency reduction over performing content routing alone, which may translate to at least tens of milliseconds of reduction in Web page load time [38]. Of course, the performance benefits from replication will depend on the replicas’ locations, but our results show that—especially for services that are difficult to replicate widely—PECAN can offer tangible benefits. In practice, performance gains will depend on the actual technique used to explore alternate network routes.

We make several contributions. First, we perform (and publicly release) millions of performance measurements over three months on a globally distributed testbed that emulates an online service provider network, to evaluate the benefits that joint content and network routing could offer to online services in practice. Although we focus on latency, we also quantify PECAN’s benefits for throughput and jitter. Second, we decompose the performance results by studying how content routing and network routing alone reduce network latency on our testbed. Finally, we have developed and deployed a prototype implementation of PECAN, which we describe in the Appendix.

2. State of the Art

We begin by providing an overview of both content routing and network routing as employed by online service providers today. We attempt to describe the state of the art, as best as we can determine through discussions with network operators and online service providers.

2.1 Global Traffic Management

Operators often refer to the system that controls how clients interact with their replicated online service as a global traffic management (GTM) system. State-of-the-art GTM systems can perform both content and network routing, but independently. Content routing refers to the process of selecting which data center replica (among a set of geographically distributed points-of-presence) should service a particular request. Network routing, in contrast, refers to the process of selecting both the wide-area, interdomain paths and the intra-domain paths within an OSP backbone that each replica will use to communicate with clients.

Figure 1 shows an example GTM system which has several components involved in directing traffic. The example shows a system that can perform content routing with both DNS-based direction and proxy-based redirection (e.g., with a front end (FE) that rewrites URLs). In this paper, we do not distinguish between the different ways that a network operator might perform content routing; we merely assume that an OSP can map clients to replicas in at least one of these ways. If the OSP operates its own backbone network with interdomain routing connections to the wide-area Internet, it can also perform network routing to adjust the paths that client traffic takes to reach each replica. For example, an OSP operator might perform network routing by adjusting intra-domain routes on the OSP backbone, or by performing interdomain traffic engineering at the border routers between the OSP backbone and the OSP’s peers.

Despite the flexibility that these GTM systems provide, OSPs may still have trouble achieving good performance for their clients, since content routing and network routing remain disjoint. On the one hand, content routing systems have limited visibility into (and control over) the alternate paths that are available to network routing systems. On the other hand, network routing systems often have relatively poor visibility into the end-to-end service performance that clients experience and how changes in network routing could improve that performance. Because content and network routing typically occur independently in today’s OSP networks, we describe each in turn below. The goal of PECAN is to bridge this gap.
2.2 Content Routing

Content routing systems have been heavily influenced by academic research, which we overview in Section 6. Today’s content routing systems perform three major functions: 1) collecting performance information, 2) mapping clients to replicas, and 3) directing clients to replicas according to the client-replica map. We describe these steps below.

Step 1: Collecting performance information. OSPs use many technologies to measure performance between online service replicas and their clients. Such measurements can be classified into active, passive, and indirect. Active measurements usually involve sending probes from replicas to clients (or v.v.), which provides direct information about the network performance to the client. Active measurements are problematic in at least two ways: 1) the probes might not be handled by network in the same way the actual traffic is handled, and 2) active measurements do not scale to a large number of replicas and customers [22]. For these reasons, in practice OSPs typically use a combination of passive and indirect measurements.

Passive measurements record the performance of the actual online service traffic between the replica and the clients to estimate the performance that clients are receiving. Passive measurements scale well and can provide direct insight into the performance of a service over the network, recording information such as TCP round-trip times and packet loss. To measure performance for all <client, replica> pairs, OSPs occasionally randomly redirect a small fraction of their clients’ requests to alternate replicas [36].

Step 2: Mapping clients to replicas. OSPs use a variety of proprietary algorithms to map clients to replicas. Client-to-replica performance is important, but there are other inputs to the algorithms that produce these mappings as well, including service availability, servicing costs, desired load, and regulatory restrictions [42].

To map clients to replicas when no active or passive measurements are available to inform selection, OSPs often resort to indirect inference of the likely performance between replicas and clients. Commercial IP geo-location databases [28] are often augmented with historical information to estimate performance between replicas and clients. As new clients begin to use the system, the OSP can update its performance estimates with passive measurements.

Step 3: Directing clients to replicas. OSPs use three main techniques to implement their client-to-replica mapping: 1) DNS-based redirection, 2) HTTP redirection, and 3) client-tailored HTML rewriting. DNS mapping uses DNS servers to respond to clients with IP addresses of best replicas. (“Clients” in this case most often are the DNS resolvers resolving names on behalf of the end-systems.) DNS mapping is most useful to improve the performance of initial resource requests. HTTP redirection can further redirect clients to a better replica (at a cost of initial request latency.) When the requested resource has multiple sub-components (e.g., a Web page with images), the OSP can use client-tailored HTML rewriting to direct clients to retrieve sub-components from disparate replicas.

Taken together, these methods allow online service operators to achieve good client performance while spreading load across their infrastructure. Yet, these content routing mechanisms operate only on the paths that are already chosen by the network operators. Next, we describe a set of popular methods that network operators use to select the paths between clients and replicas.

2.3 Network Routing

Industry has taken two main approaches towards network routing: 1) deploying commercial platforms and services for multi-homed enterprises; and 2) performing in-house traffic engineering by adjusting network configurations in the OSP backbone network. We discuss each of these approaches below.

Commercial platforms and services. Avaya and Cisco offer the PathControl [34] and Optimized Edge Routing (OER) [17] route management platforms, respectively. These platforms perform continual performance measurements to online services and adjust interdomain routes between the services and their clients based on these performance measurements. Similarly, Internap provides route optimization services for their clients [23] by performing measurements along alternate paths and redirecting traffic between services and clients by adjusting interdomain routing policy. These platforms and services primarily target large enterprises with multiple upstream providers. PECAN applies similar types of interdomain route control to adjust routes between clients and replicas, and it is possible that some variant of these systems could be used to implement aspects of PECAN’s network routing subsystem. Our evaluation also hints at how these services might scale to large OSPs who have have millions of clients and many replicas.

Conventional traffic engineering. Large OSPs primarily improve the performance of their connectivity by deploying their own networks and increasing the richness of their peering connections to improve wide-area network performance. Peering significantly reduces transit costs for large ISPs, but it can sometimes harm performance. For example, Zhang et al. showed that by preferring peering connections, Microsoft’s users on average would experience an RTT of 66 ms—more than twice the default routing policy which gives an average RTT of 29 ms [45]. This anomaly indicates that peering alone is not always effective as performance enhancing strategy.

Despite the benefits of Internet route control as highlighted in many research studies (see Section 6), the network operators that we interviewed at large OSPs rarely use anything more sophisticated than general routing policies (e.g., setting per-link BGP localpref parameters or export policies). In 2003, Feamster et al. found that wide-area routing heuristics used in ISP backbones were primitive and
Table 1: Average improvement to latency (RTT), throughput (BW), and jitter. The baseline, over which improvement is measured for each technique, is the average performance to the single best replica.

<table>
<thead>
<tr>
<th>Routing type</th>
<th>RTT (%)</th>
<th>BW (%)</th>
<th>Jitter (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best replica</td>
<td>107.35%</td>
<td>212.47</td>
<td>5.95%</td>
</tr>
<tr>
<td>Network</td>
<td>4.35%</td>
<td>0.87%</td>
<td>9.32%</td>
</tr>
<tr>
<td>Content</td>
<td>16.75%</td>
<td>8.11%</td>
<td>11.82%</td>
</tr>
<tr>
<td>Joint</td>
<td>20.44%</td>
<td>11.29%</td>
<td>17.57%</td>
</tr>
</tbody>
</table>

Figure 2: Latency reduction as a function of the number of replicas; the baseline performance is the average performance to a single best replica.

Table 2: Marginal gains of joint routing over content routing. Clients in each percentile improve by at least the amount indicated.

<table>
<thead>
<tr>
<th>Client percentile</th>
<th>RTT (%)</th>
<th>BW (%)</th>
<th>Jitter (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most-improved 5%</td>
<td>16.94</td>
<td>10.25</td>
<td>41.20</td>
</tr>
<tr>
<td>Most-improved 10%</td>
<td>13.43</td>
<td>1.42</td>
<td>29.24</td>
</tr>
<tr>
<td>Most-improved 20%</td>
<td>9.91</td>
<td>0.47</td>
<td>13.48</td>
</tr>
</tbody>
</table>

ad hoc [19], eschewing available granular routing techniques like per-prefix localpref or export policies. Our conversations indicate that the state of affairs is largely the same today: While network operators do occasionally use granular routing, they do it only in exceptional cases, when service degradation is noticeable and content routing alone cannot provide satisfactory resolution. For example, the “Why-High?” system proposed by Krishnan et al. [25] (and deployed in Google) identifies corner cases of prefixes exhibiting higher delay than expected. Systems like “WhyHigh?” can alleviate occasional problems with network routing, but they are reactive. In contrast, PECAN aims to systematically and proactively look for best performing network paths and optimize them jointly with content routing.

3. Case for Joint Content & Network Routing

We now summarize the potential gains of a joint content and network routing system; we expand on our methodology and findings in subsequent sections. We emulate an OSP setup using a globally distributed testbed that allows us to both replicate services across sites and control inbound routes to these sites. This testbed, which we describe in detail in Section 4.1, emulates an OSP with replicas in a number of geographically distinct locations with a diversity of wide-area network routes at each location.

Overview of measurements. The testbed has five replicas distributed across the United States; from each replica, we explore about 250 alternate routing choices to about 200 globally distributed clients. For six months (from July to December 2011), we collected a comprehensive <client, replica, route> performance map consisting of millions of measurements. We have released this dataset to the reviewers at an anonymous site [3] and we plan to make that data set public with publication of this work. Section 4 explains our experimental setup in more detail.

Improvement over the best replica. When OSPs roll out a new online service, it often starts at a single replica and then expands to more sites. It is interesting to know how expanding the set of replicas and/or adding joint content and network routing improves the service performance. Table 1 compares how network routing, content routing and joint routing (PECAN) each improve over a single best replica. (We formally define each metric in Section 5.)

The “network” routing row in Table 1 shows the gains if the OSP chooses to explore alternate routes only for that single best replica. Conversely, the “content” routing line shows the improvement if the OSP chooses to replicate the service to all five locations available in our testbed. Content routing provides greater performance gains than simply applying network routing for one site. Finally, the “joint” routing row shows the gains attained when the OSP chooses to both replicate the service to all five locations and perform joint content and network routing.

In practice, in addition to network-level performance, the effectiveness of both replica and network path selection depend on traffic acquisition costs, replica loads, and other variables. Unfortunately, it is hard to obtain data to model the effect of such variables. Hence, we optimize only for latency, throughput, and jitter. While this choice might bias our results, it similarly impacts both content routing and joint routing; thus, we can still compare the two.

Figure 2 shows that the benefits from joint routing are largely independent of the size of the replica set in our testbed: adding more replicas to an OSP yields latency improvements for both content and joint routing. Hence, an OSP can improve its performance using joint routing regardless of the number of replicas it currently employs. The figure shows the 80th, 85th and 90th-percentile gains over the performance of a single best replica.

Improvement over content routing. While joint routing unquestionably provides greater gains than network or con-
Table 3: The Transit Portal deployments that we use to emulate a replicated online service with route control. At each Transit Portal location, we host a replica of an online service; from each of these locations, we explore more than two hundred alternate routes (poisons) between each replica and the set of clients that it could reach.

<table>
<thead>
<tr>
<th>Service Replica</th>
<th>Location</th>
<th># of routes (poisons)</th>
<th># of measurements (RTT)</th>
<th># of measurements (BW)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Default route</td>
<td>Alternate routes</td>
</tr>
<tr>
<td>1</td>
<td>Atlanta, GA</td>
<td>259</td>
<td>292,806</td>
<td>1,455,137</td>
</tr>
<tr>
<td>2</td>
<td>Clemson, SC</td>
<td>253</td>
<td>19,401</td>
<td>1,422,832</td>
</tr>
<tr>
<td>3</td>
<td>Princeton, NJ</td>
<td>261</td>
<td>224,457</td>
<td>1,438,588</td>
</tr>
<tr>
<td>4</td>
<td>Seattle, WA</td>
<td>247</td>
<td>366,357</td>
<td>347,302</td>
</tr>
<tr>
<td>5</td>
<td>Madison, WI</td>
<td>247</td>
<td>67,473</td>
<td>1,389,266</td>
</tr>
</tbody>
</table>

4. Evaluation Setup

This section describes our evaluation methodology. We first describe the testbed infrastructure, followed by our measurement procedure.

4.1 Infrastructure

We use PlanetLab to emulate a set of clients, from which we perform measurements to the replicas over many different sets of routes, and Transit Portal to deploy an ersatz Web service with both replica and route diversity.

4.1.1 Clients: PlanetLab

We use 200 PlanetLab nodes as our client set. From the full list of PlanetLab nodes, we select nodes with which we can establish sessions. We further filter the set of these “live” nodes to include only a single node per PlanetLab site. In the end, we have a client pool with 38% of the nodes in North America, 36% in Europe, 21% in Asia, and 5% in South America.

It is well-known that PlanetLab nodes are not the best representation of the Internet. It is hard to quantify how much PlanetLab biases our measurements. On one hand, PlanetLab nodes are better provisioned and have better “last mile” connections to the immediate provider than their residential counterparts. On the other hand, we focus our measurements on the performance we can gain by exploiting replica and route diversity in the network core, and not on the network edge. Our results will be more affected by how well a PlanetLab node’s provider is connected to the Internet relative to an average Internet user. Most PlanetLab nodes’ access providers are academic institutions, whose connectivity to the Internet is often comparable to the connectivity of smaller ISPs or medium enterprises.

4.1.2 Replicas: Transit Portal

Transit Portal (TP) is a platform that enables researchers to perform experiments that require altering wide-area Internet routes [5]. There are five TP sites; each site has which is a functional Internet router, connecting to an upstream ISP, receiving a full Internet routing table. Each node is able to participate in BGP routing by issuing BGP updates from the IP address space and AS numbers allocated to Transit Portal. TP nodes allow multiple researchers to use these routing resources concurrently. We obtained access to the five TP sites described in Table 3; each of which acts as a replica in our
though it has a number of known drawbacks in practice, BGP poisoned route advertisements with path poisoning from TP sites used to control egress traffic to our replica sites. We discovered 250 different routes in a variety of ways. For example, OSPs could also preface, BGP community attributes, or route selection without requiring access to the BGP routers. BGP's loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop the BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP's AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.

BGP AS_PATH poisoning leverages the BGP-loop prevention algorithm to explore alternate routes on the Internet. As BGP route advertisements propagate through the Internet, each router attaches its own AS number to the AS_PATH attribute. BGP’s loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop a BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP’s AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.

BGP AS_PATH poisoning leverages the BGP-loop prevention algorithm to explore alternate routes on the Internet. As BGP route advertisements propagate through the Internet, each router attaches its own AS number to the AS_PATH attribute. BGP’s loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop a BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP’s AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.

BGP AS_PATH poisoning leverages the BGP-loop prevention algorithm to explore alternate routes on the Internet. As BGP route advertisements propagate through the Internet, each router attaches its own AS number to the AS_PATH attribute. BGP’s loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop a BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP’s AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.

BGP AS_PATH poisoning leverages the BGP-loop prevention algorithm to explore alternate routes on the Internet. As BGP route advertisements propagate through the Internet, each router attaches its own AS number to the AS_PATH attribute. BGP’s loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop a BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP’s AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.

BGP AS_PATH poisoning leverages the BGP-loop prevention algorithm to explore alternate routes on the Internet. As BGP route advertisements propagate through the Internet, each router attaches its own AS number to the AS_PATH attribute. BGP’s loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop a BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP’s AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.

BGP AS_PATH poisoning leverages the BGP-loop prevention algorithm to explore alternate routes on the Internet. As BGP route advertisements propagate through the Internet, each router attaches its own AS number to the AS_PATH attribute. BGP’s loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop a BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP’s AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.

BGP AS_PATH poisoning leverages the BGP-loop prevention algorithm to explore alternate routes on the Internet. As BGP route advertisements propagate through the Internet, each router attaches its own AS number to the AS_PATH attribute. BGP’s loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop a BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP’s AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.

BGP AS_PATH poisoning leverages the BGP-loop prevention algorithm to explore alternate routes on the Internet. As BGP route advertisements propagate through the Internet, each router attaches its own AS number to the AS_PATH attribute. BGP’s loop prevention algorithm, which is implemented on all BGP-speaking routers, says that a router must drop a BGP route update if the AS_PATH attribute of the route contains the AS number of said router. Dropping these updates prevents the router from accepting updates that the router has already received, thus preventing loops. As shown in Figure 4, an Internet Service Provider (ISP) can use BGP AS_PATH poisoning to exploit this algorithm by inserting a target ISP’s AS number in the AS_PATH attribute before the update is originated. The target AS, in turn, will drop the update and its clients will likely choose alternate routes to the route originator.

We use the traceroute tool to identify the ISP networks (and their AS numbers) on the default paths from our clients to each replica. We then poison these AS numbers one by one to reveal alternate paths from clients to replicas. Not every client moves to an alternate path after we issue a poisoned update: some updates affect just a few clients, while some affect a great many. To find which clients are affected by a poison we, again, use traceroute from every client to the poisoned IP prefix. There is a possibility that a some fraction of these alternate paths are a result of the wider variety of techniques at their disposal, it is plausible that OSP operators might see even greater performance gains than our experiments suggest.
As explained in the Appendix, each of our replicas advertises an un-poisoned prefix at all times. This un-poisoned prefix can always be reached to perform a measurement over the default path that rarely changes. For poisoned routes, we use the following sequence for each replica to collect a client/replica path performance map:

1. **Announce a prefix with a poisoned update.** The poison will propagate the prefix to some client networks over the alternate paths.

2. **Perform measurements to the poisoned prefix.** From every client in our client set, collect measurements to the replica using the poisoned prefix. Clients for which the poison did not affect the end-to-end path will see no improvement. Clients for which the prefix affected the end-to-end path will see either improved or reduced performance.

3. **Perform measurements to an un-poisoned prefix.** Conduct the same set of measurements over the default path (i.e., using the un-poisoned prefix) to the replica to collect a contemporaneous baseline to which we will compare our poisoned path.

As shown in Table 3, the dataset resulting from these measurements contains many more measurements of the default path than the poisoned paths. The abundance of default path measurements allows us to establish confidence in our baseline performance measurements. We consider a poisoned path between a client and a replica to improve latency over the client’s default path only if the poisoned path shows latency smaller than the minimum latency ever recorded over the default path between the client and the replica. We apply a similar litmus test for jitter measurements. For throughput, we record an improvement only if a poisoned path produced higher throughput than any throughput measurement we ever observe on a default path.

5. **Evaluation**

We evaluate the benefits of joint routing with respect to latency, throughput and jitter. We also compare how well joint routing performs compared to traditional content routing.

### 5.1 Baseline Performance

When considering the performance improvements that different routing approaches induce, we must establish a baseline to compare them against. In this section, we use two baselines for comparison: 1) a best replica baseline, and 2) a content routing baseline. Before formally defining these baseline metrics, we describe how we perform the measurements that help us establish these baselines.

**Measurements of default path performance.** Figure 5 shows the CDF of minimum latencies clients experience to each replica over a default path. The minimum latency for each client is obtained from a large set of measurements: On average, each client measures the default path to a replica 6,692 times. The figure shows two major groups of clients: About 40% of the clients have latencies between 0–50 ms, and about 60% of the clients see latencies of 90 ms or larger, with just a few in between. These modalities reflect the geographic distribution of our client dataset: About 38% of clients are in the U.S and Canada and see lower latencies, while the rest of the clients are overseas.

Similarly, Figures 6 and 7 show the CDFs of maximum throughput and the minimum jitter, respectively, as observed by clients to each replica. As with latencies, the maximums and minimums are computed over a set of measurements for
the default path between each <client, replica> tuple. For each such tuple, we have, on average, 189 jitter and throughput measurements. Figure 6 highlights why it is difficult to measure capacity using the PlanetLab nodes as clients. The clients in the figure form three distinct groups: 1) those with 10 Mbps links, 2) those with 100 Mbps links, and 3) those with speeds above 100 Mbps. The 10 Mbps and 100 Mbps groups identify cases where the PlanetLab nodes are directly connected to a bottleneck link; in the first two cases the bottleneck is at the client itself and neither joint routing nor network routing can improve throughput.

Best replica baseline. When a new online service is launched, it often starts with a single replica. We want to know how much the network performance improves over that single replica when the OSPs start adding more replicas and implement content routing or joint routing. We define the best replica for some performance metric as the replica that the largest fraction of clients would select, given that each client can select its own best replica based on that performance metric. Table 4 shows the breakdown of popularity of different replicas when each client selects a replica based on the performance of the default paths for each <client, replica> tuple. The average performance to the best replica across all clients yields the average best replica performance. For example, as shown previously in Table 1, the average latency that clients experience to the best replica (which, as indicated in Table 4, is replica 3) is 107.35 milliseconds.

Table 4: Percentage of clients for which a given replica is the best choice.

<table>
<thead>
<tr>
<th>Replica</th>
<th>Latency</th>
<th>Throughput</th>
<th>Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.93%</td>
<td>7.77%</td>
<td>12.62%</td>
</tr>
<tr>
<td>2</td>
<td>1.15%</td>
<td>35.92%</td>
<td>35.92%</td>
</tr>
<tr>
<td>3</td>
<td>56.64%</td>
<td>6.77%</td>
<td>8.74%</td>
</tr>
<tr>
<td>4</td>
<td>22.54%</td>
<td>48.54%</td>
<td>18.45%</td>
</tr>
<tr>
<td>5</td>
<td>12.71%</td>
<td>0.97%</td>
<td>24.27%</td>
</tr>
</tbody>
</table>

Content routing baseline. PECAN extends content routing to also incorporate the benefits of network routing; to quantify the additional benefit of PECAN relative to content routing, we compare the performance of PECAN against the performance that content routing alone provides. Recall that a content routing system maps each client to its own best replica. Formally, for a client $i$, the content routing latency $RTT_{content,i}$ is

$$RTT_{content,i} = \min_{j \in (1, ..., M)} \hat{RTT}_{ij0},$$

where $M$ is number of replicas and $\hat{RTT}_{ij0}$ is the minimum latency that we measured between client $i$ and replica $j$ over the default path (noted as path 0). Taking the average across all clients yields average content routing performance. In addition to showing the performance of clients to each replica, Figures 5–7 show the performance of a content routing system when each client is directed to its own best replica (labeled “content routing”). Note that the content routing system we implement uses only network performance as the basis for selecting a replica for each client; in practice, OSPs also use replica loads and costs to inform content routing decisions.

5.2 How Well Does Joint Routing Work?

We quantify the benefit that PECAN provides when compared to content routing. When PECAN is in use, we formally define client’s $i$ latency as

$$RTT_{PECAN,i} = \min_{j \in (1, ..., M)} \min_{l \in (0, ..., K_{ij})} \hat{RTT}_{ijl},$$

where $M$ is the number of replicas, $K_{ij}$ is the number of paths between client $i$ and replica $j$, and $\hat{RTT}_{ijl}$ is the minimum latency we recorded over the path $l$ between client $i$ and replica $j$. Recall that path $l = 0$ corresponds to the default path. With measurements of both content routing and PECAN performance, we can compute the percentage improvement as $100 \cdot (RTT_{content,i} - RTT_{PECAN,i})/RTT_{content,i}$. We use the same approach for jitter; for throughput, we take maximums instead of minimums. Below we provide a breakdown of the average performance improvements that we presented in Table 1 from Section 3.

Latency. Joint routing delivers an additional 4.3% (or about 5 ms) round-trip latency reduction on average beyond performing content routing alone, which may translate to significant improvements in service response times [38]. Figure 8 shows the percentage improvement in latency over the content routing baseline. The solid line in the figure shows improvement for all clients, while the dashed line shows shows the latency reduction for clients that had a baseline latency of 0–50 ms. We find that 20% of our clients see a reduction in latency of at least 10%. We also observe that clients with baseline latencies of 0–50 ms see similar improvements, with 18% of these clients improving by 10% or more. This result is significant, since content replication can only reduce latency by placing content closer to the
users. At some point, however, placing replicas close to all clients might become prohibitively expensive; in such cases, an OSP might rely on PECAN to improve routing between the closest replica and the clients nearby.

As discussed earlier, Figure 2 plots content and joint routing benefit over the best replica baseline as we increase the number of replicas. Adding replicas provides higher improvements with joint routing, but with decreasing marginal improvement at every addition. Content routing behaves similarly, albeit with a lower improvement at each step.

**Throughput and Jitter.** When compared to the baseline of content routing, only 5% of clients that are using PECAN experience a throughput improvement of 20% or more, while almost 90% of clients see no improvement. We attribute this limited improvement to the inability of most PlanetLab nodes to saturate their Internet path bottlenecks. In case of jitter, 10% of the clients see approximately 30% less jitter, while 60% of clients see no improvement at all. As with throughput, we observe that PlanetLab nodes’ inability to saturate router buffers along Internet paths limits the observed jitter.

## 5.3 Why Does Joint Routing Work?

Joint routing improves performance over content routing alone because it provides multiple alternate network paths for each replica. As explained in Section 4, PECAN finds about 3.4 alternate paths on average for each <client, replica> tuple. Even when a client cannot improve its performance by switching replicas, network routing can often improve performance to one of the replicas. Figure 9 shows latency improvements from network routing alone for each replica. For each replica, the baseline over which we compute the latency reduction is the minimum latency over the default path to that replica. We find that 20% of the clients experience improvements of 5–20%, depending on which replica we choose to evaluate. Some replicas (e.g., Replicas 1 and 2) have relatively poor default paths, so approximately 80% of clients achieve some benefit from alternate paths to these replicas.

## 5.4 Scalability and Stability

To be practical, PECAN must judiciously limit the number of route changes it broadcasts to the Internet; it also should limit oscillations induced by large numbers of clients changing replicas. In this section we seek to assess these requirements by analyzing two questions: (1) How many routes must the OSP explore to achieve the benefits of joint routing? And (2) how many clients change their preferred replica after joint routing is applied?

### 5.4.1 Scaling route selection

OSPs that tweak egress routes can do so without affecting the Internet routing system. To explore the alternate ingress routes, however, OSPs must issue additional routing updates. All of the experiments presented so far evaluate the improvements that PECAN can provide by using 250 ingress routes (achieved via poisoned advertisements) at each geographic location. Over time, OSPs might be able to evaluate all of these 250 configurations, but doing so frequently may not be practical. Hence, we consider how many routes an OSP needs to explore to see improvement from joint content and network routing.

When an OSP uses a limited set of routes to feed traf-
fic to virtual replicas, it must decide which routes to use, but selecting the optimal subset of routes for each replica is computationally intractable. To avoid exploring all possible route combinations of route advertisements from each replica, we devise a simple heuristic: for each replica, we order the routes based on the average improvement they provide when compared to the performance over the default path. We then take the routes sequentially form that order and announce them from the replica. For example, if OSP decides to announce three of the alternate routes, it will pick three top routes from the ordered list. We compare this heuristic against selecting sets of routes at random.

Figure 10 shows how performance improves as we increase the number of announced routes. The gains in the figure are shown over the best replica baseline. The figure contains four plots: “maximum”, “ordered”, “random”, and “content”. The “maximum” line represents the maximum gain that an OSP can get with joint routing (see Table 1). The “ordered” line shows the gains as we increase number of alternate routes from 0 to 16. The routes here are picked using the heuristic described above. The “random” line shows the gains when we add additional routes at random. To generate the “random” plot, we run 10 iterations and report average and standard error values. Finally, the “content” line shows the content routing gains over the best replica (also from Table 1). In most cases, the ordered heuristic outperforms random route selection. It is also encouraging that only five additional virtual replicas per physical replica can obtain approximately 60% of the performance gains that are possible with all 250 alternate routes.

5.4.2 Stability of replica selection

Content providers typically take loads at each replica into account when assigning clients to replicas. With PECAN, however, clients that previously had a suboptimal replica will shift their traffic to other replicas. This shifting might adversely affect demand on replicas, causing overloads. In fact, such imbalances could even increase end-to-end service latency. The effect of joint routing on the stability of current replica choices is thus a genuine concern.

If deploying PECAN on an existing content routing system creates such imbalances, its usage would not be practical: a provider would have to entirely reconfigure a replica setup to account for the change in traffic and load. Fortunately, we find that, when optimizing for latency, for 93% of the clients the choice of replica due to content routing does not change during joint routing. This suggests that PECAN is stable enough for practical deployment, since the loads on each replica will not change significantly with a small number of moves.

6. Related Work

In this section, we review related work in improving content routing and network routing.

6.1 Content Routing

The late 1990s witnessed the first efforts in optimizing mapping of end-users to content or service replicas. Bhat-tacharjee et al. [12] presented a seminal paper describing a client-to-replica mapping system. This system used IP anycast to reach a directory service (e.g., DNS) which then routed the clients to the best service replica based on the performance map. Seshan et al. [36] invented a new way to collect a comprehensive <client, replica> performance map: a small fraction of clients would be directed to randomly selected replicas to estimate the performance. Andrews et al. [11] presented a system called Webmapper that showcased algorithms for performing approximate client-to-best-replica matching.

The initial step in client to replica mapping is usually performed using the Domain Name System (DNS). Pang et al. [32] evaluate the responsiveness of DNS to changes in client-to-replica mapping and find that in many cases DNS is sluggish to respond. Nonetheless, DNS is still the primary method for directing initial client requests to the best replicas. Huang et al. [22] introduced a DNS reflection method for client-to-replica performance map generation. Instead of usual client traffic, DNS reflection forces Local DNS (LDNS) servers to use iterative queries to remote replicas to estimate the delay between the LDNS servers and the replicas. The LDNS performance information is then used as a proxy metric for client-to-replica performance. Most recently, Wendell et al. [42] described a system called DONAR that allows authoritative DNS servers to make mapping decisions with only partial global information.

6.2 Network Routing

We first discuss content routing techniques that require changes either in the end-systems or in significant parts of the Internet routers. Despite the benefits of such systems, they are still to see universal deployment. Then, we describe proposals that can operate within the constraints of today’s networks and protocols.

Overlay Routing and Clean-Slate Network Routing Proposals. In the late 1990s, the overlay networks were a popular research topic. Informed Internet Routing and Transport Savage et al. [35] explored the benefits of an overlay routing system that selects best performing alternative paths. In addition to a conventional routing system underneath, the system requires an active network of overlay nodes to improve user experience. Andersen et al. [10] deployed and evaluated a similar overlay system across diverse locations in the Internet; commercial content distribution networks ultimately applied many of these techniques for finding the best paths to pre-cached content [30].

The mid 2000s saw several proposals to improve routing in unconventional ways. Yang et al. [44] presented a system that can increase path diversity with routing deflections. End hosts in such systems can set bits that instructs routers on the path to perform deflections over better paths. Xu and Rex-
ford [43] introduced MIRO: a system that provides increased diversity of paths choices for interdomain routing. Motiwala et al. [31] presented a routing algorithm for Internet routers that enables scalable exploration of Internet path diversity. Unfortunately, utilizing such systems requires changes in Internet routers and in end-hosts.

**Improvements to Conventional Internet Routing.** In the last decade there has been a lot of research on wide-area routing, most of it focusing on the effectiveness of multihoming enterprise networks. Our work builds upon these efforts and extends them to include scenarios where an OSP has a choice not only of diverse network paths (network routing) but also a choice of replicas (content routing). For example, Akella et al. [6–9] explored the effects of multihoming on the performance of a site that either sends or receives Internet traffic. The authors study 68 Akamai nodes in 17 cities as a testbed: Same city often contains multiple nodes, each with a different upstream ISP; the authors connect to all the Akamai nodes in one city to estimate performance of each ISP in that city, effectively emulating a multihoming setup in that city. The authors found that route optimization produces greater benefits in peak time intervals. Unfortunately, the study was limited in the number of alternative Internet paths (only upstreams) and it did not extend network routing to a logical conclusion of joint network and content routing.

Goldenberg et al. [20] assessed the benefit of single-site multihoming and also considered cost in the analysis. Guo et al. [21] analyzed a commercial solution for multihomed enterprises, which focused on possible performance gain with two upstream ISPs. Lee et al. [26] explored ways to scale active measurements for multihomed enterprises. Uhlig et al. [40] and Wang et al. [41] proposed formalizing upstream ISP selection as an optimization problem. Most of the efforts mentioned above focus on the enterprise setting and do not compare content routing with network routing.

7. Limitations and Future Work

In this section, we discuss some limitations of our current study and directions for future work. We focus in particular on how our dataset might be made more representative.

**Size of the testbed.** One of the most serious challenges in evaluating the performance gains than an OSPs can attain is to have a replica set that matches that of real OSPs. This equivalence entails two primary components: (1) diversity of replica locations and (2) diversity of route choice in each location. In terms of geographic diversity, our set of replicas is comparable to the set of North American Amazon EC2 data centers, although it is much smaller than the infrastructure of a large commercial OSP such as Google or Microsoft. In future work, we plan to perform similar experiments with replicas hosted across a Tier-1 ISP backbone network; we are in the process of deploying this measurement infrastruc-

**Route exploration technique.** Our method limits us to exploring the routing diversity at each of our replicas that BGP AS_PATH poisoning can provide. Moreover, poisoning introduces undesirable routing churn. In practice, however, OSPs have a much wider variety of techniques at their disposal for controlling both ingress and egress routes. Most OSPs can choose among multiple egress routes to destinations and control ingress routing by selective announcements, BGP community attributes, or BGP AS_PATH prepending. The fact that real OSPs have more expressive route control techniques suggests that they may be able to realize even greater performance gains than we observed.

**Client set.** Another challenge in emulating real-world OSP performance is obtaining a representative client set. In our case the clients are PlanetLab nodes, which are hardly a representative set of Internet end hosts. Many of the nodes we use are housed in well-connected university campuses. It does bias our client set, but it is not clear whether performance improvements—especially latency improvements—would differ with a more representative set of clients. On one hand, PlanetLab nodes might be better connected than average Internet nodes, providing greater route diversity to and from such nodes. On the other hand, PlanetLab nodes might be better-provisioned in general than typical end hosts, and, thus, hard to improve on. Less well-connected and more remote networks might see more significant performance improvements from joint routing.

**Measurement method.** A future study might also attempt to measure or approximate the overall user experience of using a particular replica and network route, perhaps approximating user experience by page load times, as has been done in previous work on Web performance [39]. Because our clients are run from PlanetLab nodes, it is not practical to instrument a browser and record the performance from each client. A promising direction for future work would be to conduct a more comprehensive study of how systems such as PECAN can improve user-perceived performance.

**PECAN deployment costs.** To put the performance improvement PECAN achieves into perspective, one must measure the cost needed to build a functioning PECAN system such as the one described in the Appendix. These costs depend on the ease of integration between content routing systems and inter-domain traffic engineering systems. Although existing content routing systems, such as those described in Section 2, are highly programmable, the same cannot be said about today’s routers. Recent advances in Software Defined Networking [29] indicate that this type of programmatic integration might be less costly in the future.

8. Conclusion

Online service providers currently perform replica selection (content routing) and route selection (network routing) independently. We introduce PECAN (Performance En-
hancements with Content And Network routing), a system that performs joint content and network routing for an OSP that wishes to improve end-to-end performance to clients. We design PECAN as an extension to the content routing systems that OSPs currently use.

We evaluate the performance of PECAN on a globally distributed testbed that emulates a modern OSP by running the replicated online service on five Transit Portal (TP) sites, each offering a large choice of network paths to clients. We use 200 PlanetLab nodes as clients to estimate network performance to our replicated service. Our experiments show that PECAN reduces latency by 4.3% (or nearly 5 ms) on average over simply performing content routing alone, which may translate to at least tens of milliseconds of reduction in Web page load time [38]. Finally, we find that PECAN can provide most of the potential benefit with only a few judiciously selected network paths: exploring just five sets of alternate network routes between clients and replicas in our testbed can yield 60% of the maximum possible benefit of joint routing to an online service provider. Given the increasing reliance of today’s online services on even small improvements in latency, the improvements that PECAN yields over standard content routing may be warranted for certain latency-sensitive services, particularly in cases when replication itself is costly.

REFERENCES

Figure 11: Content and network routing subsystems allocate isolated resources to explore new network routes.

APPENDIX

A. Implementing PECAN in Practice

As we described in Section 2, modern OSPs use sophisticated content routing systems to load balance requests between replicated data centers in an attempt to improve client performance. In this section, we will describe how operators can extend their existing content routing systems to support network routing as well. In particular, we present the design of PECAN (Performance Enhancements with Content And Network routing), a system that enables seamless integration of content and network routing.

A.1 Exposing Routing Choices

The basic idea behind PECAN is to extend an OSP’s current <client, replica> mapping infrastructure to instead map clients to <replica, route> pairs. To do so, PECAN breaks each replica into a set of virtual replicas, where each virtual replica corresponds to a different choice of routes to the replica (i.e., a single <replica, route> tuple). Figure 11 shows how PECAN allows a content routing system to tap into network route diversity. The router in the figure has a separate routing slice dedicated to each set of alternate routes (virtual replica). For example, in today’s routers such a slice can be implemented using Virtual Routing and Forwarding instances (VRF) or a variety of alternative technologies.

A.1.1 Egress routes

There are a wide variety of mechanisms available to employ alternate egress routes from a given virtual replica. For example, conventional BGP multihoming can increase route diversity; operators can use BGP’s local preference setting to adjust the choice of egress routes to each client prefix. PECAN could also benefit from protocols such as Detour [35], RON [10], Platypus [33], Deflections [44], and Path Splicing [31], all of which increase an end-systems choice of (and control over) egress paths. Similarly, considering industry proposals, many practitioners see Locator/ID Separation Protocol (LISP) [18] as a feasible improvement to BGP. LISP separates the Endpoint Identifier (EID) (i.e., a host IP address) information from Routing Locator (RLOC) (i.e., the information that encodes the location of the EID in the wide-area Internet.) LISP could allow an OSP to explore potential egress routes by selecting the entry points to a remote network as encoded with RLOCs.

A.1.2 Ingress routes

Affecting a virtual replica’s ingress routes is more challenging since it requires changing the way other networks forward packets. The key to enabling distinct route sets for each virtual replica is to separate these routing decisions. In PECAN, an OSP allocates a distinct IP address prefix to each virtual replica. Hence, to map clients to a particular virtual replica, PECAN need only point them to an IP address within the virtual replica’s prefix.

Today’s Internet supports a number of ways to impact route selection, including selective prefix announcement (i.e., announcing a prefix only to a subset of neighbors), prepending AS_PATH attributes, setting BGP communities or MED attributes, and BGP AS_PATH poisoning; we evaluate employing AS path poisoning in PECAN extensively in the next section. Future technologies, such as LISP, might provide even more elegant alternatives.

Crucially, by maintaining one virtual replica (address prefix) at each physical replica that always uses the default network paths, PECAN does no harm: clients can always obtain the performance provided by content routing alone if none of the joint routing options provide superior performance.

A.2 Selecting a Virtual Replica

We now describe how PECAN’s virtual replicas enable the process of joint content and network routing. The joint optimization could happen in many ways; we take an iterative approach, as shown in in Figure 12. First, PECAN optimizes network routing between each client/replica pair:
1. Enumerate the route options.
   - Provision ingress/egress routing
   - Create virtual replicas

Remove under-performing virtual replicas

2. Select best virtual replicas.
   - Collect performance metrics
   - Compute <client, virtual replica> map

Update performance metrics

3. Direct clients to virtual replicas.
   - DNS and HTTP redirects
   - Client-tailored HTML re-writing

Figure 12: Joint network and content routing selection with PECAN.

for each replica, PECAN identifies the network path to each client that yields the best performance, and establishes a virtual replica with that path preference. Then, for each client, PECAN selects the virtual replica that offers the best performance among the available options at each physical replica.

This process proceeds in three steps, which could be either automated or manually performed by the operators of the network and online service.

1. **Enumerate the route options.** OSP network operators must enumerate the alternate routes from each replica to clients that the system should explore. Depending on the route selection technique employed, the operator may wish to enumerate egress (e.g., a choice of a next-hop neighbor) routes and ingress (e.g., selective route announcement) routes separately, or jointly. Evaluating all possible alternate routes to each client is unlikely to scale, but our evaluation (Section 5) shows considering just five virtual replicas (i.e., sets of alternate ingress routes) at each replica can realize performance improvements that are 60% of the maximum possible improvement.

2. **Select the best virtual replica for each client.** PECAN evaluates the performance of each virtual replica for each client. To evaluate a new virtual replica (route selection) for a given client and physical replica, PECAN redirects a small fraction of client requests to the virtual replicas and evaluates the performance that the client sees. PECAN gradually increases the number of clients mapped to a virtual replica to avoid overloading any network path or physical replica. Isolating test measurements from the bulk of the traffic requires a set of dedicated load-balancing proxies, as shown in Figure 11. As long as the evaluated route offers improved performance for enough clients and is reliable over the test period, PECAN maintains the virtual replica in the set of virtual replicas that can be used for joint routing.

3. **Direct clients to virtual replicas.** Once PECAN has selected the best virtual replica for each client, it implements the mapping. To implement this mapping, PECAN uses DNS load balancing to map each client to a virtual replica IP address, where the BGP prefix for that IP address corresponds to the route that the PECAN has selected for that client and replica using the previous steps. Because PECAN maps each virtual replica to its own prefix, a client always has the option of using either default content routing (i.e., the route in today’s CDN) or the PECAN-provided route.