On the Effectiveness of Measurement Reuse for Performance-Based Detouring

David Choffnes and Fabián E. Bustamante
Department of Electrical Engineering & Computer Science, Northwestern University
{drchoffnes,fabianb}@cs.northwestern.edu

Abstract— **For both technological and economic reasons, the default path between two end systems in the wide-area Internet can be suboptimal. This observation has motivated a number of systems that attempt to improve reliability and performance by routing over one or more hops in an overlay. Most of the proposed solutions, however, fall at an extreme in the cost-performance trade-off. While some provide near-optimal performance with an unscalable measurement overhead, others avoid measurement when selecting routes around network failures but make no attempt to optimize performance.**

This paper presents an experimental evaluation of an alternative approach to scalable, performance detouring based on the strategic reuse of measurements from other large distributed systems, namely content distribution networks (CDNs). By relying on CDN redirections as hints on network conditions, higher performance paths are readily found with little overhead and no active network measurement. We report results from a study of more than 13,700 paths between 170 widely-distributed hosts over a three-week period, showing the advantages of this approach compared to alternative solutions. We demonstrate the practicality of our approach by implementing an FTP suite that uses our publicly available SideStep library to take advantage of these alternative Internet routes.

I. Introduction

Path selection in the wide-area Internet is known to be suboptimal in terms of end-to-end latency, loss rate and TCP throughput. Building on a large body of previous work measuring the behavior of Internet routing, the Detour study showed the potential benefits of detouring flows via a third node [31]. Since then, there has been a number of proposed overlay routing systems that attempt to improve reliability and performance [5], [14], [30], [36]. Most solutions fall at either extreme in the cost-performance trade-off: RON provides near-optimal performance at the cost of measurement overhead that is quadratic in the number of nodes in the system, whereas Gummadi et al.'s technique requires no measurement overhead to route around network failures, but does not attempt to optimize performance.

In this paper, we present an experimental evaluation of an alternative approach to scalable, performance detouring. The proposed approach relies on the strategic reuse of measurements from other large-scale distributed systems, namely content distribution networks (CDNs). CDNs cache copies of web objects on thousands of servers worldwide and redirect clients to different servers, over short time scales, based on server load and network conditions [2]. In previous work [35], we demonstrated that these redirections are primarily driven by network conditions and suggested that they could be used to identify quality Internet paths without additional monitoring. This paper presents a thorough evaluation and comparison of *SideStep*, a detouring service we have implemented based on these ideas.

SideStep employs CDN redirections as hints on network conditions, to efficiently identify higher performance paths with little overhead and no active network measurement. Paraphrasing Lampson [18], a hint is the saved result of some measurement or computation used for the purpose of making a system run more efficiently. Since hints may be wrong, there must be a way to check their correctness before taking any unrecoverable action. SideStep employs an effective, low-cost strategy for validating the quality of recommended paths.

Our work makes the following contributions:

- A detailed description of the design and implementation of the **SideStep detouring service**. SideStep is the first open-access, scalable solution to finding high-quality overlay paths.
- **Results from a wide-area evaluation** of the deployed system, proving that CDN redirection dynamics can be seen as hints regarding high-quality candidate detour points, and that these hints can effectively support a highly scalable detouring service.
- An **open-source API and library** implementing our SideStep detouring service, along with an FTP suite (**DraFTP**) that relies on SideStep to seamlessly take advantage of alternative Internet routes and serves as a model for other client applications.

Before describing our experimental approach and presenting our evaluation results, we briefly describe the design and implementation of SideStep (Sec. II). We then experimentally show the benefit of the SideStep detouring service in terms of end-to-end performance improvements (Sec. IV). Our results show how reusing CDN measurements allow us to eliminate the scalability constraint imposed by actively measuring all overlay paths (e.g., as done in the RON approach). Finally, we demonstrate the practicality of our approach by implementing *DraFTP* – an FTP suite that uses our portable, publicly available SideStep library to seamlessly take advantage of alternative Internet routes. We discuss the limitations of our approach and challenges for future work in Sec. V and conclude in Sec. VII.

II. BACKGROUND AND RELATED WORK

To the best of our knowledge, ours is the first experimental evaluation of the effectiveness of measurement reuse for performance detouring.

There has been a number of proposed overlay routing systems that attempt to improve reliability and performance. Early approaches to reliable overlay networks (RONs) require extensive monitoring that scales with the square of the number of nodes in the system and thus limits their scope to small deployments (10s of nodes) [5]. A number of related efforts have investigated alternative techniques for path selection to address the problem of measurement overhead in overlay systems. Proposed approaches vary from exploiting AS-level path information [12] or building on a common routing underlay dedicated to topology probing [23] to relying on passive measurements at end hosts [32] or opportunistically combining passive measurement of wide-area service traffic with targeted active probing [38].

More recently, Gummadi et al. [14] demonstrated that a system can recover from a majority of interior network failures [11] without such overhead by picking a random relay point. Their approach, however, does not focus on improving performance—in our own experiments, selecting detour points at random improves end-to-end throughput significantly (by over 10%) only 11% of the time.

Similar to RON and Detour, and unlike Gummadi et al., our approach focuses on *improving* end-to-end throughput between two Internet hosts. It differs from RON and Detour in that it avoids *additional* probing overhead by reusing measurements performed by other long-running services to locate detour points. SureRoute [3] (also known as AkaRouting) is a private detouring service sold commercially by Akamai. It is a closed, proprietary system that, like RON, uses extensive network measurements to find high quality overlay paths. Our service does *not* use paths provided by SureRoute.

In [35], we reported on a broad measurement study of the Akamai CDN and demonstrated that their redirections are performed frequently enough as to be useful for control, that these updates are primarily driven by network conditions and are, therefore, potentially beneficial to other applications. Our early ping-based study illustrated the potential benefits of employing CDN redirections for identifying good detouring paths and demonstrated that in approximately 50% of scenarios, the best measured one-hop path through an Akamai server outperforms the direct path in term of latency. Subsequent work has shown that CDN information can be used for other purposes such as accurate network positioning without the cost of active network measurement [34] and biased neighbor selection in P2P systems [9].

This paper extends our previous work in three significant ways. First, in [35] we measure one-hop paths through servers from the Akamai CDN, which are not available to an independent overlay network. In contrast, *this work evaluates detour paths through nodes that participate in our service and uses redirection information from multiple CDNs*. Second, the previous work evaluates path *latencies* over synthetic paths, while this work evaluates *end-to-end throughput over real, complete paths*. Finally, *we built a real system and deployed an example application* (DraFTP) that uses CDN-based hints to locate and route traffic through high quality detour points that improve end-to-end throughput.

Our approach achieves high scalability by employing CDN redirection dynamics as hints to achieve performance gains with low overhead. Hints are a well established and widely adopted technique in systems. They are significantly less expensive to maintain than facts (i.e., observations based on direct measurement) and are able to improve system performance when accurate. Lampson [18] reports on the use of hints in operating systems, networking, languages and applications. Hints have also been successfully employed in other contexts, from file systems [24], [29] and memory management [7] to web caching [22].

SideStep is part of a research effort driven by the observation that a large fraction of wide-area systems can be built to ensure sustainable scalability by strategically reusing the view of the network gathered by long-running, pervasive services such as CDNs. These pervasive services can act as oracles for other systems [1], ensuring that the latter scalability comes without imposing unduly large loads on underlying shared resources. Part of this work focuses on developing efficient techniques to match available network information, gathered at low cost from existing oracles, with the needs of distributed systems. Thus, we see our work as complementary to the ongoing recent projects that have begun to address the challenges in supporting Clark et al.'s [10] grand vision of a knowledge plane for supporting large-scale, self-managing distributed systems [13], [20], [27], [37].

III. SIDESTEP DESIGN

SideStep is designed to improve performance for data transfers between two endpoints. Specifically, the goal of our service is to locate and detour data streams across overlay paths that improve performance in terms of end-to-end throughput.

The SideStep service provides client applications access to an I/O stream interface that transparently forwards data along alternative paths to improve performance. SideStep identifies potential quality detour paths by employing CDN redirection dynamics to locate a set of candidate detour points, collectively referred to as *detour groups*. While SideStep is running, it periodically performs DNS translations on CDN names (i.e., URLs) to update its redirection dynamics, makes a summary of this information available via a distributed hash table (DHT) and reads a list of summary information from other nodes. Nodes with similar redirection behavior are assigned to the same detour group and thus become candidates for detouring.

Because CDNs redirections provide only *hints* regarding network conditions, SideStep must validate those hints before redirecting the entire data flow over the corresponding detour paths. SideStep does this by splitting the data stream between candidate detour paths and the current path, then comparing each path's throughput as reported by the destination. *The* *winner of this "race" is the path over which the entire data stream is sent.* As we discuss in Section II-B, splitting the stream allows us to evaluate candidate detour paths without incurring any end-to-end throughput penalty.

The following paragraphs provide additional details on the architecture of SideStep and discuss our main design choices.

A. Managing Redirection Dynamics

CDNs attempt to improve web performance by delivering content to end users from multiple, geographically dispersed servers located at the edge of the network [2], [19], [21]. Content providers contract with CDNs to host and distribute their content. Since most CDNs have servers in ISP points of presence, clients' requests can be dynamically forwarded, via DNS redirections or URL rewriting, to topologically proximate replicas [16], [33]. Beyond static information such as geographic location and network connectivity, CDNs rely on network measurement subsystems to incorporate dynamic network information in replica selection and determine high-speed Internet paths over which to transfer content within the network. Our early ping-based study shown that in approximately 50% of scenarios, the best measured one-hop path through an *Akamai server* outperforms the direct path in terms of *latency*.

We thus use CDN redirections dynamics as hints, based on previous work demonstrating that two nodes exhibiting similar redirection behavior are very likely to be along high-quality paths to one another and are thus good candidate detour points for each other [9], [35]. While our prior work showed how to encode redirection behavior and compare them *offline* [34], here we introduce a scalable approach to managing and distributing redirection dynamics *online*.

We encode redirection behavior as a map of ratios, where each ratio represents the frequency with which the node has been directed toward the corresponding replica server during the past time window. Specifically, if node N_a is redirected toward replica server r_1 30% of the time and toward replica server r_2 70% of the time, then the corresponding ratio map is:

$$\nu_a = \langle r_1 \Rightarrow 0.3, r_2 \Rightarrow 0.7 \rangle$$

To determine whether two nodes a and b are mapped to the same detour group, we compute the *cosine similarity* of their redirection maps, which returns a value between 0 and 1. In particular, for a given threshold t, if $cos\_sim(a,b) \geq t$, then hosts a and b are in the same detour group.[1]

Before comparing two ratio maps, our system must first be able to locate nodes' ratio maps in a scalable and efficient manner. We note that ratio-map information is naturally organized as key-value pairs: a ratio map is tied to a node identifier (e.g., a node's IP address) and each ratio-map entry ties a replica server to the frequency with which it is witnessed. Given this structure, a DHT (which stores data as key-value pairs) is a natural and scalable solution for storing and retrieving such information. We discuss the details of SideStep's DHT-based technique for storing mapping data in Section IV.

[1]A complete description of the encoding/comparison is found in [34].

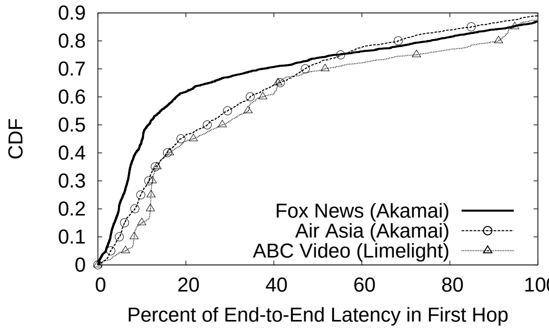


Fig. 1. Ratio of first-hop to end-to-end latency over all CDN-recommended paths, demonstrating that different CDN customers lead to different detour-group characteristics.

Quality detour paths should exhibit good path characteristics along each hop and follow Internet routes that are significantly different from the direct one. Unfortunately, CDN-based detour groups will not provide these properties if only a small number of clients are served by a particular replica server, and if these clients are in the same ISP.

We address this issue by exploiting the fact that CDNs offer differentiated levels of service to their customers. For example, consider Akamai customers CNN (an American news corporation) and Air Asia (an airline serving Asia). Using ratio maps gathered from lookups to the CNN domain name, two nodes in Illinois appear in a different detour group than two nodes in Nebraska (the groups are ≈ 450 miles apart). However, using Air Asia, these four nodes are in the same detour group. Thus, by using different CDN customers, we can access a more diverse set of detour paths.

Figure 1 demonstrates this property by comparing the RTT latency to a node in one detour group (the first hop of a detour path) with the end-to-end latency for the direct path to a node outside the group. The figure plots the cumulative distribution function (CDF) of three curves using all of the detour paths found by SideStep; the x-value of a point on each curve represents the ratio of the first-hop latency to the end-to-end latency. The figure clearly shows that for CDN customer Fox News, detour group nodes are on average much closer to each other than for CDN customers Air Asia and ABC Video. For example, 52% of the detour points found using the Fox News customer name exhibit RTT latencies to nodes inside the detour group that are at least eight times smaller than the latency to nodes outside. For the Air Asia customer name, however, only 34% of the detour group nodes provide the same characteristics. Due to this level of diversity in redirection behavior, SideStep maintains separate ratio maps for each CDN customer, and compares ratio maps only between the same customer.

While different CDN customer names form detour groups with diverse path characteristics, we show in Section IV-B that all of the CDN names used in the study can improve performance for clients. Thus, to maintain scalability under heavy load, SideStep can spread detouring traffic over multiple detour groups by having nodes use different CDN names.

B. Validating Detour Paths

After mapping overlay nodes to CDN-based detour groups, we must validate the hint that group members are good candidate detour points. This means we must determine whether one-hop overlay paths through nodes in the CDN detour group offer higher end-to-end throughput than the direct path.

We validate candidate detour paths by "racing" them; i.e., by spliting the data stream over each path, and comparing the throughput of each path as perceived by the destination host.

The evaluation period lasts until throughput along each path has stabilized or a predefined number of bytes has been sent along the detour path. The first condition ensures that both TCP flows achieve a steady state before their performance is compared while the latter one guarantees that races are of finite duration (if path characteristics lead to high TCP throughput variability). When the race has completed, the source uses only the best path, based on the throughput as reported by the destination.

Because we interleave client data over the two paths being compared, there is *no end-to-end throughput penalty* for evaluating each path. The potential cost of this approach comes from the additional delay that can be imposed by a poor detour route and the corresponding software complexity to handle out-of-order packet delivery at the destination when reassembling the stream from two different paths. In practice this overhead is significant only if detouring occurs near the end of the stream and thus increases time-to-completion. In our experiments, we detected no decrease in performance due to races.

It has been pointed that most of the performance gains for detouring can be achieved using only one overlay hop [5] as done in SideStep. As part of our future work, we intend to investigate the potential benefits of two-hop detouring in the context of SideStep.

IV. Evaluation Setup

The goal of our evaluation is to examine SideStep performance over time and across a variety of geographic regions. We begin by describing our measurement methodology and presenting details of the SideStep implementation.

Recall that SideStep relies on CDN redirection dynamics as hints for making detouring decisions, and we have shown that these dynamics respond to real-time changes in network conditions [35]. Logically then, the best approach to evaluate SideStep is through a widely-deployed, experimental testbed such as PlanetLab [26].

During a three-week period (April, 2008), we evaluated the effectiveness of our system in terms of finding high quality detour paths between 318 distinct source–destination pairs from 170 PlanetLab nodes. We used at most one endpoint per PlanetLab site, with 48% of the nodes in North America, 40% in Europe, 8% in Asia, 2% in South America and 1 node in Oceania. While the reported results are, naturally, specific to our experimental testbed and the particular time of our experiments, we believe they indicate trends that are likely to continue in other SideStep deployments.

Parameter	**Value**
DNS lookup frequency	1 hour
Ratio map expiration	24 hours
Cosine similarity threshold, t	0.2
Race frequency	60 s
Maximum race data	3 MB
Minimum throughput gain to switch paths	5%
Throughput-drop to trigger a race	33%
Direct-path probe frequency	300 s

TABLE I
SIDESTEP DEFAULT PARAMETERS.

SideStep Implementation: The implementation used in our evaluation relies on the Akamai and Limelight CDNs for hints. Akamai boasts the largest CDN deployment and thus generally offers the best opportunities to form CDN-based detour groups. We use a fixed set of Akamai customer names (e.g., `a1921.g.akamai.net`, which corresponds to CNN) as sources for CDN mappings, though this set can be dynamically updated. For the Limelight CDN, which similarly enjoys a global deployment, we used the domain name associated with a video-on-demand site for a popular US television network.

When performing races among candidate detour paths, SideStep monitors the throughput along each path and terminates the race when the variation in the time-averaged throughput is sufficiently small or when the maximum allowed amount of data has been sent along the path—whichever comes first. Recall that since data flows are sent in parallel, the total throughput between the endpoints generally stays the same or increases, but does not decrease during a race. Thus, besides the potential (and small) cost of delay, SideStep does not negatively affect the data transfer when validating hints.

When the race terminates, the destination reports the throughput over each path to the sender. If the improvement in throughput over the new path is above a certain threshold, the system switches to the new path. Since best path—be it a detour path or the direct path—may change during data transfer, SideStep performs races both periodically and dynamically in response to sudden changes in throughput along the current path.

SideStep offers a number of configurable parameters to control its operation, such as the CDN names that are used and the frequency with which races are performed. For most of our experiments, we used a set of values that that have shown to work well in practice, which are presented in Table I. Due to space limitations, we focus our evaluation on determining the quality of CDN hints for detouring and the relative performance of using different CDN customers.

V. Experimental Results

In this section, we present results from our evaluation of SideStep. We focus on SideStep's performance and overhead.

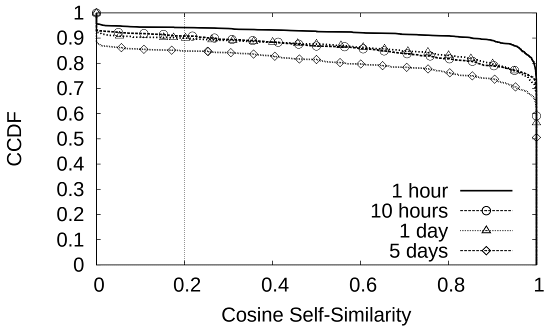


Fig. 2. Cosine self-similarity over various time scales, indicating that DNS lookups can be performed as infrequently as once per hour.

A. SideStep Overhead

There are two primary components to SideStep's technique for locating detour points: obtaining redirection dynamics and using the DHT to read and write summary redirection information. We now analyze their overheads.

Obtaining Redirection Dynamics: To determine how infrequently DNS lookups can be performed without loss of accuracy in terms of replica-server mappings, we analyzed ratio-map information generated by running SideStep on PlanetLab nodes. Ideally, we would like ratio maps to be relatively stable over short time scales, to reduce the frequency with which name translations are performed. On the other hand, we would like ratio maps to be sufficiently dynamic as to be responsive to changing network conditions.

We use the cosine similarity metric to monitor changes to the same node's ratio map over time. A cosine-similarity of 1 means that CDN redirection dynamics did not change at all during the observed time period, while a cosine-similarity value of 0 indicates that redirection dynamics have changed enough to place the node in a different detour group.

Figure 2 plots the complementary CDF (CCDF) of the cosine-similarity values, so a point (x, y) means that y percent of samples have a cosine-similarity value greater than x for a given curve. Each curve represents a different time window in the set of 1 hour, 10 hours, 1 day and 5 days. We use a cosine-similarity threshold of 0.2 for detour-group membership. The closer the curve to the top of the graph, the more stable the mapping. For example, during the course of 1 hour, over 80% of our sample of ratio maps did not change at all; i.e., their cosine similarity values are 1. At the other extreme (on the same curve), only 1% of the ratio maps changed detour groups.

The figure also shows that cosine similarity values tend to decrease as the time interval between DNS lookups increases; i.e., nodes' ratio maps can change significantly over the time scale of hours and days. Thus, while ratio maps could be updated as infrequently as once per hour, updating them less frequently can lead to significant loss in accuracy.

Locating Detour Points: We use the cosine similarity of two nodes' ratio maps to determine whether they belong to the same detour group (Sec. II-A). If cosine similarity is high, the nodes are tightly bound to the same detour group.

To facilitate comparisons, SideStep currently supports storing and retrieving ratio maps from a service DHT [28] via a generic interface. After each ratio update, SideStep places the node's current ratio data in the DHT using the node's listening socket address as the key.[2] To enable nodes with similar ratio maps to find this content in the DHT, the node also creates a reverse mapping by adding its socket address to the DHT, using each *strongly mapped* replica server (cluster) as a key. A node is considered strongly mapped to a replica server if the percent of time it has been seen is greater than or equal to the cosine similarity threshold. In this way, we never store information about mappings that are not useful for identifying detour group members.

Since a node's ratio map can change depending on the CDN customer associated with the DNS lookup, SideStep maintains a separate ratio map for each CDN customer. When searching for nodes in the same detour group, SideStep compares two nodes' ratio maps only if they are from the same customer.

Because CDNs often colocate multiple servers in ISPs' PoPs, both for load balancing and redundancy, nodes are often directed toward multiple replica servers in the same class-C subnet, i.e., having IP addresses that differ only in the last quartet. In this case,[3] we cluster all servers in the same class-C subnet and maintain one entry for the cluster in our ratio map. This has the additional benefit of significantly reducing the amount of overhead required to store mappings (from 6,246 unique replica-server IP addresses to a set of 879 clusters), without any loss of accuracy in terms of the quality of hints.

The overhead required to maintain and distribute mapping information is quite low. To maintain mapping information in the DHT, a node publishes its ratio values each time they significantly change (e.g., once per hour), then publishes its socket address using each *frequently seen* replica-server cluster as a key. In our study of mapping behavior, we have found that nodes see a small set of replica-server clusters (< 10) very frequently and see others much less so. Thus, as a rule of thumb, nodes publish mapping information only for replica servers to which they are redirected more than a fraction t of the time. (Recall that t is the cosine-similarity threshold, a number less than one.) Consequently, publishing mapping information requires at most $1 + c/t$ writes, where c is the number of CDN customers used. The first term occurs because the node uses one write operation to store its significant ratio-map information using its socket address as a key. The c/t term occurs because a node can have at most $1/t$ entries in its ratio map with a value greater than or equal to t. For each such entry, the node must create a reverse mapping by adding its socket address to the value stored at the key for that entry. Thus, in the worst case, if the ratio-map entries for each CDN customer are orthogonal and there are $1/t$ entries per customer, then there will be c/t writes.

Retrieving information from the DHT incurs a similarly small overhead – in fact, at most two lookups are required

[2]The listening socket address identifies the node's IP address and port used for incoming SideStep data connections.

[3]Noteworthy exceptions are servers with IP addresses owned by the CDNs.

before SideStep can begin detouring. We achieve this lower bound because SideStep performs lookups using an asynchronous, iterative process, allowing a node to begin exploring detour paths as soon as it retrieves a single detour path from the DHT. Although the total number of DHT operations performed to lookup all nodes in the same detour group ($n * c/t$) scales linearly with the average number of nodes, n, mapped to each replica-server cluster, the maximum number of lookups that SideStep performs in practice is lower and depends on the duration of the associated file transfer.

B. Cumulative Results

In this section, we present cumulative results from our experiments in which over 13,700 paths were evaluated using two approaches: picking detouring points at random and using SideStep. Experimenting with randomly-selected paths has the property of sampling *all* paths given enough time. Though this leads to prohibitive wasted throughput in deployment, it permits us to approximate the best detour path available to the system. We also note that picking detour paths at random was shown by Gummadi et al. [14] as a way to route around path failures; in this section we evaluate the extent to which it can be used for improving performance over a direct path that has not failed.

Experiments are configured as follows. At the beginning of an experiment, the source node connects to the destination and begins transferring data containing random bytes. After a brief warm-up period, the source node starts evaluating detour paths. If the experiment is configured to use a random set of detour points, middle hops are picked at random with equal probability; otherwise middle hops are picked according to our description of SideStep in the previous section. We used 100 MB file transfers—a size that generally provided sufficient time to evaluate at least one detour path and not so large so as to exceed PlanetLab's daily bandwidth limits.

We conducted file-transfer experiments to characterize what type of source–destination pairs can benefit from detouring. Those that cannot find any candidate detour paths are excluded from our study because we cannot compare relative performance with detouring. Fortunately, the percent of nodes in this category is small — 14%.

The actual performance that a detouring system sees is strongly dependent on the the *best* detouring path that is found during a transfer, since lower performance ones would be abandoned after conducting a race. Note that the best "Random" path is an approximation to the "optimal" path because given enough time, Random tries *all* paths available to the system. To analyze relative performance for the best detouring paths, we use the best Random performance as a baseline and label it "best known". We then evaluate two practical alternatives to trying all paths at Random: trying at most k paths either randomly or from SideStep.

Figure 3 plots the difference in transfer rates between the direct path and the best known paths during an experiment. The graph shows that picking one detour path at random works only about one third of the time, while picking one from

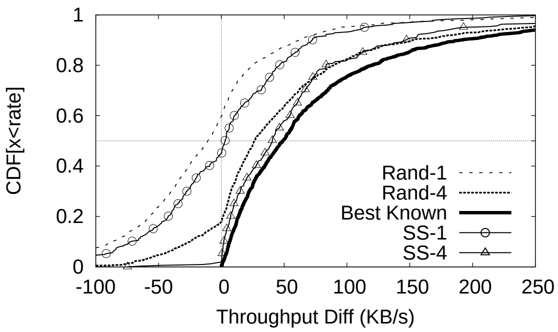


Fig. 3. CDF of differences between transfer rates along the direct path and the best detour path for each experiment.

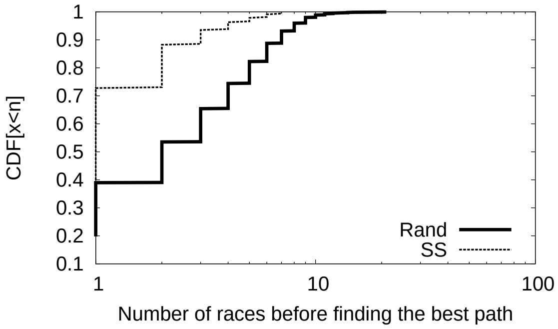


Fig. 4. CDF of the number of detour paths that are evaluated before finding the best one.

SideStep improves performance about half of the time. When the detouring system is allowed to pick at most four paths, the performance improves, with SideStep providing near-optimal performance most of the time.

An important issue for heuristic-based detouring systems is how many paths are evaluated before finding detour path that improves performance. The more paths that must be evaluated, the higher the overhead (due to race traffic) and the less likely it is that the system will improve performance before the transfer completes. In Fig. 4, we plot the number of detour paths explored by a random algorithm and SideStep before finding the best detour path. The graph shows that SideStep locates the best path on the first try over 70% of the time, whereas Random does so less than 40% of the time. Further, 90% of the time SideStep finds the best detour path within the first 3 tries whereas the random approach requires three times as many tries to do the same. Thus, not only does SideStep yield better performance than Random on average, it does so with significantly less racing bandwidth and time.

While the previous figures focused on performance using *any* CDN customer name for SideStep, we now investigate the relative difference for performance for each individual customer name. In Fig. 5, we plot the differences between transfer rates along the direct path and the average rates along the detour path. There is one curve for each CDN customer name, and we include the random curve as a baseline. Here, the Akamai-based paths offer the best average performance,

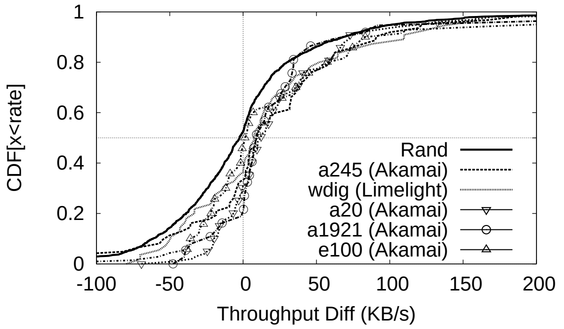


Fig. 5. CDF of differences between transfer rates along the direct path and the average rates for all of the detour paths attempted for each customer name.

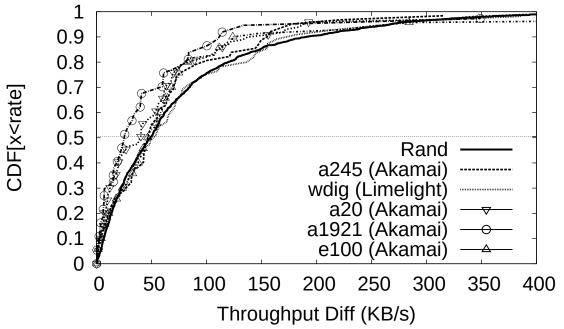


Fig. 6. CDF of differences between transfer rates along the direct path and the best detour path for each SideStep customer name.

while Limelight tends to offer the lowest (but still better than random). We believe the reason for lower average performance using the Limelight name is that the corresponding CDN uses smaller numbers of replica server data centers worldwide, resulting in larger detour groups. This larger detour group results in larger numbers of candidate detour paths, but not a proportionately larger number of high-quality ones.

In Fig. 6, we plot the difference between the direct path and the best detour path performance, for each CDN customer name used in this study. As a baseline for comparison, we include the best known paths found by randomly selecting paths, which represents the "best known" path as described earlier. We see that the Limelight CDN offers the best maximum performance gains, while the different names for the Akamai CDN tend to offer lower, but similar performance. Again, we believe that the larger detour groups for the Limelight CDN increases the number of candidate detour paths and thus provides more opportunities for finding the best detour path. As shown in the previous graph, though Limelight allows SideStep to find better maximum performance, its average performance is lower due to a larger number of low-quality detour paths that it finds.

In parallel with our experiments, we performed and recorded `traceroute` measurements for each path that streamed data, at the beginning of each experiment and at the beginning of races. (Note that `traceroute` is not part of the SideStep service and was only used here for the purpose of evaluation.) When conducting measurements over detour paths, the source node performs a `traceroute` to the detour node, the detour node performs a `traceroute` measurement to the destination node, and we later compose the two paths for analysis. We used this information to compare basic path characteristics between the default Internet path and those found by SideStep.

First, we use these measurements to compare end-to-end latency between the default Internet path and the one based on CDN hints. Figure 7(a) visualizes this information using a CDF of the *ratio* of direct-path latency to the detour-path latency. The figure shows that approximately 65% of the one-hop paths recommended by SideStep result in lower latency than the direct path. Further, nearly half (48%) of the detour paths reduce latency by 10% or more.

We also analyzed our data to determine what portion of the direct path was avoided by the detour path, indicating the diversity in Internet routes. Figure 7(b) illustrates this metric using a CDF plot.

The figure shows, for example, that detour paths always differ from direct ones in at least 8% of the path while 65% of the paths differ in at least half of the total hops taken by the direct path. Thus, the majority of hops along paths found by SideStep are different than those on the direct path, for most of the samples. In short, it is clear that SideStep does find diverse Internet paths; moreover, as Fig. 1 shows, the majority of these paths are along quality paths in terms of latency.

Next, we use the `traceroute` data to compare loss rates along the alternate Internet paths. Figure 7(c) presents a CDF of the *difference* in loss rates between the direct path and the CDN-based alternate path, over all the experiments. We compute these loss rates by determining the percentage of `traceroute` measurements dropped by routers during the entire measurement.

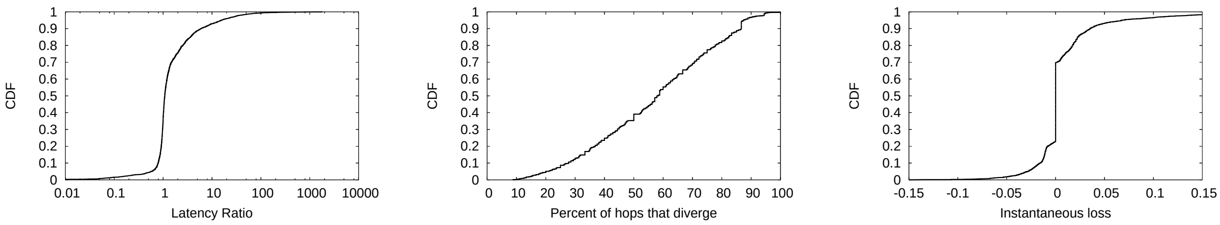
The figure clearly shows that loss rates along the detour path are as low or lower than the direct path more than 75% of the time. More importantly, the detour path reduces loss over the direct path more than 30% of the time. Thus, in the majority of cases, SideStep finds high-quality detour paths in terms of instantaneous path loss.

C. SideStep FTP Suite

To demonstrate the broad applicability of our approach, we implemented an FTP suite that uses our publically available SideStep library.

SideStep is packaged as a library that can run as a standalone service to provide detouring capabilities to participating peers. It also contains an API with four basic calls for client applications: two for registering an incoming or outgoing data connection and two for requesting an input or output TCP stream. The current SideStep implementation is written in Java for cross-platform portability and contains approximately 7,600 LOC.

We modified an existing open-source FTP suite to use our detouring service. We found that integrating SideStep into the FTP client and server was fairly straightforward. The server required changes to 27 lines of code, while the client required



(a) Ratio of direct-path to detour-path latencies; nearly half of detour paths significantly improve end-to-end latency.

(b) Divergence between direct and detour paths; SideStep finds detour paths offering significant path diversity.

(c) Difference in loss rates between direct and detour path; majority of SideStep detour paths improve loss rates.

Fig. 7. Characteristics for paths found using SideStep.

changes to only 10 lines of code. The reason for the small amount of code is that SideStep provides access to detouring via the commonly used I/O stream interface. Thus, modifying the FTP code required changing only the source of the stream to one provided by SideStep instead of the default network library.

The SideStep FTP suite, DraFTP, is available publicly online[4] under a free, open-source license. We intend for it to serve as a model for how to modify other data-transfer software to use our service. In addition, SideStep is running constantly on various PlanetLab nodes, providing a number of "seed" nodes for future use and further experimentation with the service.

VI. Discussion and Future Work

This work presents a detailed evaluation of measurement reuse for performance-based detouring. In this section, we detail the limits of this approach, discuss issues that must be addressed should it enjoy large-scale adoption and comment on the strategic reuse of CDN information.

Limits of the Approach: Our experiments were conducted in PlanetLab. The advantage of this approach is that we can demonstrate the effectiveness of our system on a large variety of real Internet paths. The disadvantage is that PlanetLab nodes may suffer from unusually high loads, leading to unpredictable (and often large) application-layer delays and variations in available throughput. The reported results showed that our approach can work even in this hostile environment. A topic of future work, however, is to determine how information regarding load and available bandwidth can be incorporated in detouring decisions.

SideStep is currently designed to improve performance for bulk TCP data transfers. If the entire file can be transferred along the direct path in less time than is required to find a good alternate path (that take longer than one minute on average (in our experiments, on the order of 10s of seconds), then SideStep cannot provide any benefit to the file transfer.

Our study showed that a large majority of alternate paths located by SideStep offered lower latency than the direct path. An interesting direction for future work is investigating how well SideStep can use these paths to provide performance benefits for low-bandwidth, latency-sensitive applications.

[4]http://www.aqualab.cs.northwestern.edu/projects/SideStep.html

Large-scale Deployments: There are many issues that must be addressed should SideStep be adopted by a large number of hosts relative to our experimental testbed. We focus on several key issues below.

For one, it is possible that a detour path has enough *total* bandwidth to improve end-to-end throughput for an incoming flow, but does not have enough *available* bandwidth for that flow. To address this issue, nodes can exchange information about their current throughput conditions during the connection-establishment handshake. Specifically, the source node should send the candidate detour node an estimate of its current throughput along the direct path. The middle node can likewise passively monitor its maximum throughput and use that to estimate its available bandwidth. Based on this information, it can simply reject the detour connection if there is not enough available bandwidth to serve the new flow.

Given a large-scale adoption, one should address the issue of fairness in terms of bandwidth consumption among peers. In particular, a peer is consuming bandwidth unfairly if it sends a large amount of data traffic over detour paths but does not carry any detour traffic for other data flows. Should this become a significant problem in terms of reducing available bandwidth in the SideStep system, we expect that a credit-based mechanism (e.g., [15]) can provide incentive for peers to contribute their their fair share of bandwidth to the system.

Another concern with large-scale deployment is the issue of authentication and access control. This is important, for example, when peers in the SideStep network form a private detouring overlay. SideStep should also be resilient to malicious behavior, such as DoS attacks or corruption of DHT information used for locating detour nodes. We leave these important issues as part of our future work.

On the reuse of CDNs' network views: It is important to note that our detouring technique does *not* place a large (or even significant) burden on the CDNs from which the system gathers network information. Because our system queries its local DNS server to determine replica-server mappings, DNS lookups can be answered from the local DNS cache without contacting the CDNs' DNS servers. Further, because our system performs only name translations and does not actually download CDN content, there is no additional data-traffic load placed on the CDN servers. Finally, we demonstrated that mappings between nodes and replica servers are stable over time scales as long as one hour. Thus, the load that our system places on the DNS infrastructure can be as low as 24 name translations per day—likely a vanishingly small fraction of those generated by web clients running in the same network.

Finally, while SideStep employs CDN redirections in previously unanticipated ways, it is important to note that our system's interactions with CDNs in no way forces them to behave in ways that contradict their fundamental policies. Further, the Akamai CDN provides summary information about live, global network conditions on their public website for free [4]. Because SideStep places an insignificant load on CDNs and accesses information already explicitly provided at no charge to the public, we expect a commensalistic relationship between SideStep and CDNs.

VII. Conclusions

This paper experimentally demonstrated the effectiveness of measurement reuse for performance-based detouring. It showed that the SideStep detouring service, which relies on CDN redirections as hints on network conditions, finds higher performance paths with little overhead and no active network measurement. Our extensive wide-area measurements evaluating more than 13,700 paths between 170 widely-distributed hosts over a three-week period, showed that SideStep can quickly find near-optimal paths when selecting from CDN-based detour points. For those paths that do not improve performance, we shown that a simple technique can efficiently evaluate the validity of CDN-based hints. Finally, we showed that our approach is practical by implementing an FTP suite that uses our publicly available SideStep library to seamlessly take advantage of these improved Internet routes.

References

[1] Aggarwal, V., Feldmann, A., and Scheideler, C. Can ISPs anda P2P users cooperate for improved performance. *ACM SIGCOMM Computer Communication Review 37*, 3 (2007), 29–40.

[2] Akamai. Akamai CDN. http://www.akamai.com.

[3] Akamai. Sureroute, May 2003. http://www.akamai.com/dl/feature_sheets/fs_edgesuite_sureroute.pdf.

[4] Akamai. Akamai introduces first-of-its-kind, real-time view into health of the Internet, June 2007. http://www.akamai.com/html/about/press/releases/2007/press_060707.html.

[5] Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R. Resilient overlay networks. In *Proc. of the ACM SOSP* (Oct. 2001).

[6] Bornstein, C., Canfield, T., and Miller, G. Overlay routing networks (Akarouting), 2002. http://www-math.mit.edu/steng/18.996/lecture9.ps.

[7] Brown, A. D., and Mowry, T. C. Taming the memory hogs: using compiler-inserted released to manage physical memory intelligently. In *Proc. of USENIX OSDI* (October 2000).

[8] Chinoy, B. Dynamics of Internet routing information. In *Proc. of ACM SIGCOMM* (September 1993).

[9] Choffnes, D. R., and Bustamante, F. E. Taming the torrent: A practical approach to reducing cross-ISP traffic in P2P systems. In *Proc. of ACM SIGCOMM* (2008).

[10] Clark, D. D., Partridge, C., Ramming, J. C., and Wroclawski, J. T. A knowledge plane for the Internet. In *Proc. of ACM SIGCOMM* (August 2003).

[11] Dahlin, M., Chandra, B. B. V., Gao, L., and Nayate, A. End-to-end WAN service availability. *IEEE/ACM Transactions on Networking 11*, 2 (April 2003).

[12] Fei, T., Tao, S., Gao, L., and Guérin, R. How to select a good alternate pah in large peer-to-peer systems. In *Proc. of IEEE INFOCOM* (April 2006).

[13] Gibbons, P., Karp, B., Ke, Y., Nath, S., and Seshan, S. IrisNet: an architecture for a world-wide sensor web. *IEEE Pervasive Computing 2*, 4 (2003).

[14] Gummadi, K., Madhyastha, H. V., Gribble, S. D., Levy, H., and Wetherall, D. Improving the reliability of Internet paths with one-hop source routing. In *Proc. of USENIX OSDI* (December 2004).

[15] Gupta, M., Judge, P., and Ammar, M. A reputation system for peer-to-peer networks. In *Proc. of NOSSDAV* (2003), ACM Press.

[16] Kangasharju, J., Ross, K., and Roberts, J. Performance evaluation of redirection schemes in content distribution networks. *Computer Communications 24*, 2 (2001), 207–214.

[17] Labovitz, C., Malan, G. R., and Jahanian, F. Origins of Internet routing instability. In *Proc. of IEEE INFOCOM* (March 1999).

[18] Lampson, B. W. Hints for computer system design. In *Proc. of the ACM SOSP* (October 1983).

[19] LimeLight Networks. Limelight networks CDN. http://www.limelightnetworks.com.

[20] Madhyastha, H. V., Isdal, T., michael Piatek, Dixon, C., Anderson, T., Kirshnamurthy, A., and Venkataramani, A. iPlane: an information plane for distributed systems. In *Proc. of USENIX OSDI* (November 2006).

[21] Mirror Image. Mirror image CDN. http://www.mirror-image.net.

[22] Mogul, J. C. Hinted caching in the web. In *Proc. of the ACM SIGOPS European Workshop* (September 1996).

[23] Nakao, A., Peterson, L., and Bavier, A. A routing underlay for overlay networks. In *Proc. of ACM SIGCOMM* (August 2003).

[24] Patterson, R. H., Gibson, G. A., Ginting, E., Stodolsky, D., and Zelenka, J. Informed prefetching and caching. In *Proc. of the ACM SOSP* (December 1995).

[25] Paxson, V. End-to-end routing behavior in the Internet. In *Proc. of ACM SIGCOMM* (August 1996).

[26] PlanetLab. Planetlab: An open testbed for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/.

[27] R, H., Hellerstein, J., Boona, N., Loo, T., Shenker, S., and Stoica, I. Querying the Internet with PIER. In *Proc. of VLDB* (September 2003).

[28] Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I., and Yu, H. OpenDHT: a public DHT service and its uses. In *Proc. of ACM SIGCOMM* (2005).

[29] Sarkar, P., and Hartman, J. Efficient cooperative caching using hints. In *Proc. of USENIX OSDI* (October 1996).

[30] Savage, S., Anderson, T., Aggarwal, A., Becker, D., Cardwell, N., Collins, A., Hoffman, E., Snell, J., Vahdat, A., Voelker, G., and Zahorjan, J. Detour: Informed Internet routing and transport. *IEEE Micro 19*, 1 (January-February 1999), 50–59.

[31] Savage, S., Collins, A., Hoffman, E., Snell, J., and Anderson, T. The end-to-end effects of Internet path selection. In *Proc. of ACM SIGCOMM*.

[32] Seshan, S., Stemm, M., and Katz, R. H. SPAND: shared passive network performance discovery. In *Proc. of USENIX USITS* (December 1997).

[33] Shaikh, A., Tewari, R., and Agrawal, M. On the effectiveness of DNS-based server selection. In *Proc. of IEEE INFOCOM* (April 2001).

[34] Su, A.-J., Choffnes, D., Bustamante, F. E., and Kuzmanovic, A. Relative network positioning via CDN redirections. In *Proc. of the Int'l Conference on Distributed Computing Systems (ICDCS)* (2008).

[35] Su, A.-J., Choffnes, D. R., Kuzmanovic, A., and Bustamante, F. E. Drafting behind Akamai: Travelocity-based detouring. In *Proc. of ACM SIGCOMM* (September 2006).

[36] Tang, C., and McKinley, P. A distributed multipath computation framework for overlay network applications. Tech. rep., Michigan State University, 2004.

[37] Wawrzoniak, M., Peterson, L., and Roscoe, T. Sophia: An information plane for networked systems. In *Proc. of HotNets* (November 2003).

[38] Zhang, M., Zhang, C., Pai, V., Peterson, L., and Wang, R. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *Proc. of USENIX OSDI* (December 2004).