NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-07-08
October 9, 2007

SideStep – An Open, Scalable Detouring Service

David R. Choffnes and Fabián E. Bustamante

Abstract

For both technological and economic reasons, the default path between two end systems in the wide-area Internet can be suboptimal. This 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.

We present SideStep, the first globally scalable detouring service that provides high-performance detour paths with a small, constant measurement overhead per node. SideStep's efficiency comes from its strategic reuse of measurements from other large-scale distributed systems --- namely content distribution networks (CDNs). Our approach
is to use CDN redirections as hints (i.e. potentially inexact information used to improve a system's efficiency) on network conditions. By carefully observing these redirections, higher performance paths can readily be found with little overhead and no active network measurement.

We present results from an evaluation of more than 9,000 paths between 133 widely-distributed hosts over a six-week period. In our study, we find that over half of the
alternative paths identified through CDN redirections yielded performance improvements over the default one. Of those, 75% resulted in noticeable TCP throughput improvements while over 11% more than doubled throughput as compared to the direct path. Finally, we demonstrate the practicality of our approach by implementing an FTP suite that uses our publicly available SideStep library to seamlessly take advantage of these improved Internet routes.

Keywords: Detouring, CDN, Performance, Networking, Measurement Reuse

SideStep – An Open, Scalable Detouring Service

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 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.

We present SideStep, the first globally scalable detouring service that provides high-performance detour paths with a small, constant measurement overhead per node. SideStep's efficiency comes from its strategic reuse of measurements from other large-scale distributed systems — namely content distribution networks (CDNs). Our approach is to use CDN redirections as hints (i.e. potentially inexact information used to improve a system's efficiency) on network conditions. By carefully observing these redirections, higher performance paths can readily be found with little overhead and no active network measurement.

We present results from an evaluation of more than 9,000 paths between 133 widely-distributed hosts over a six-week period. In our study, we find that over half of the alternative paths identified through CDN redirections yielded performance improvements over the default one. Of those, 75% resulted in noticeable TCP throughput improvements while over 11% more than doubled throughput as compared to the direct path. Finally, we demonstrate the practicality of our approach by implementing an FTP suite that uses our publicly available SideStep library to seamlessly take advantage of these improved Internet routes.

1 Introduction

Building on a large body of previous work measuring the behavior of Internet routing, the Detour study showed that Internet path selection is not generally optimal in terms of end-to-end latency, loss rate and TCP throughput [32].

Since then, there has been a number of proposed overlay routing systems that attempt to improve reliability and performance [5, 13, 31, 37]. 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. *We present SideStep, the first globally scalable detouring service that provides high-performance detour paths with a small, constant measurement overhead per node.*

SideStep achieves high scalability by reusing the network views gathered by content distribution networks (CDNs) as part of their normal operation. 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 [36], we demonstrated that these redirections are primarily driven by network conditions and posit they could be used to identify quality Internet paths without additional monitoring. In this paper, we present the design and implementation and report on a thorough performance evaluation of a detouring service based on these ideas.

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.

We first describe the design and implementation of the SideStep detouring service. We demonstrate how CDN-based hints, obtained with low overhead via infrequent DNS translations, allow us to eliminate the scalability constraint imposed by actively measuring all overlay paths (e.g., as done in the RON approach). Paraphrasing Lampson [18], a hint is the saved result of some measurement

or computation used for the purpose of making the 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 finding out the truth about recommended paths.

We then experimentally show the benefit of the SideStep detouring service in terms of end-to-end performance improvements when comparing CDN-based detour paths to the default ones. Our results are drawn from our evaluation of more than 9,000 paths between 133 widely-distributed hosts during a six-week period. Over 53% of the alternative paths identified through CDN redirections yielded performance improvements over the default one. Of those, 75% resulted in TCP throughput improving by over 10%, while nearly 11% more than doubled the throughput of the direct path. In addition to evaluating the throughput, we performed `traceroute` measurements for every path, which we use to explore the relationship between the observed throughput improvements and basic path characteristics.

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. The implementation of DraFTP required changing fewer than 40 lines of code from an existing FTP suite.

The key contributions of this paper are:

- A detailed description of the design and implementation of the SideStep detouring service, an example service based on strategic measurement reuse. 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 SideStep API and library implementing our detouring service, along with an FTP suite that relies on SideStep to seamlessly take advantage of alternative Internet routes and serves as a model for other client applications.

After reviewing backgroung and related work in the following section, we describe SideStep design and implementation in Sec. 3 and Sec. 4, and report our experimental results in Sec. 5. We discuss the limitations of our approach and challenges for future work in Sec 6 and conclude in Sec. 7.

2 Background and Related Work

To the best of our knowledge, SideStep is the first open-access, performance-oriented detouring service to achieve high scalability. SideStep builds on prior efforts in the area of detouring, CDN behavior and CDN-based systems.

Following a large body of previous work measuring the behavior of Internet routing [8, 16, 17, 25], the Detour study showed that Internet path selection is not generally optimal in terms of end-to-end latency, loss rate and TCP throughput [32]. Since then, there has been a number of proposed overlay routing systems that attempt to improve reliability and performance [5, 13, 31, 37].

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]. More recently, Gummadi et al. [13] demonstrate that a system can recover from a majority of interior network failures [10] without such overhead by picking a random relay point. This approach, however, does not focus on improving performance—in our own experiments, picking detour points at random improves end-to-end throughput significantly (by at least 10%) only 11% of the time. Similar to RON and Detour, and unlike Gummadi et al., SideStep focuses on *improving* end-to-end throughput between two Internet hosts. SideStep differs from RON and Detour in that SideStep avoids *additional* probing overhead by reusing measurements performed by other long-running services to locate its 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. SideStep is a public, free service that uses CDN redirection dynamics as hints for locating detour paths. Our service does *not* use paths provided by SureRoute.

A number of related efforts have investigated alternative approaches for path selection to address the problem of measurement overhead in overlay systems. Proposed approaches vary from exploiting AS-level path information [11] or building on a common routing underlay dedicated to topology probing [22] to relying on passive measurements at end hosts [33] or opportunistically combining passive measurement of wide-area service traffic with targeted active probing [39]. More generally, a number of recent projects have begun to address some of the challenges in supporting Clark et al.'s [9] grand vision of a knowledge plane for supporting large-scale, self-managing distributed systems [12, 19, 27, 38]. SideStep provides an approach that is complementary to these proposals by reusing information gathered by CDNs about the network and applying this information to drive a detouring service. Similar to several of them, SideStep provides this service without requiring any new infrastructure.

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, 20, 23]. 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 [15, 34].

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 [6]. In [36], 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 Akamai one-hop path outperforms the direct path in term of latency.

SideStep extends our previous work in three significant ways. First, the above work measured one-hop paths through servers from the Akamai CDN, which are not available to an independent overlay network. In contrast, SideStep 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. In fact, as we show Section 5.2, we found that for the evaluated paths, latency and throughput are only weakly correlated. Finally, we built a real system and deploy 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.

SideStep achieves high scalability by employing CDN redirection dynamics as hints to achieve high performance 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, 30] and memory management [7] to web caching [21].

3 Design

The goal of the SideStep service is to locate and detour data streams across overlay paths that improve performance in terms of end-to-end throughput. These high-quality detour paths should exhibit good path characteristics along each hop and provide access to improved performance by following Internet routes that are significantly different from the direct one.

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*. In this section, we show that when different nodes exhibit similar CDN redirection dynamics, they tend to be along high-quality paths to one another. Further, in Section 5.2 we demonstrate that they provide sufficient path diversity to realize performance benefits from 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. As we discuss in Section 3.4, splitting the stream allows us to evaluate candidate detour paths without incurring any end-to-end throughput penalty.

The following sections describe the architecture of SideStep and discuss our main design choices. Before diving into the details, we provide a concrete example of how our system works using a real detour path that our service found when streaming data from UC Berkeley to Dartmouth College.

3.1 Example SideStep Transfer

Figure 1 illustrates a transfer between UC Berkely (UCB) and Dartmouth College (DC) geographically. In this example, the source node at UC Berkeley and a node at Intel Research Berkeley (IRB) were frequently redirected to the same set of CDN replica servers at California State University. Thus, our system mapped the nodes UCB and IRB to the same detour group and used this fact as a hint that IRB would be a good candidate detour node. Subsequently, SideStep validated this hint by splitting the data stream between the one-hop overlay path UCB—IRB—DC to determine whether it would improve performance. In fact, the resulting throughput increased by approximately 33% along the one-hop path.

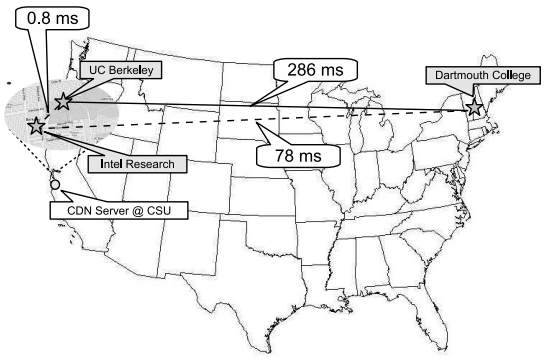


Figure 1: Geographic depiction of SideStep detouring. In this example, SideStep found a path that reduced latency by a factor of four, leading to throughput increasing by 33%.

To understand the relationship between throughput improvements and basic paths characteristics, we conducted `traceroute` measurements along both paths during this experiment. We found that the average latency along UCB—IRB—DC (0.8 ms + 78 ms) was almost four times less than that along the direct path UCB—DC (286 ms). Further, we found that even though the first overlay hop of the detour path took less than 1 ms, the resulting overlay

path was completely different from the direct path, with the exception of routers inside the UCB and DC networks. In summary, SideStep avoided active path monitoring by using CDN redirection dynamics to locate a high-quailty candidate detour point that significantly improved end-to-end performance.

3.2 Architecture

The SideStep service architecture, illustrated in Figure 2, is fairly straightforward. Client code (e.g., an FTP client or server) registers itself with the detouring service by specifying the endpoint for the connection before requesting an input stream and/or output stream for network I/O. While SideStep is running, it periodically performs DNS translations on CDN names (i.e., URLs) to update its redirection dynamics (handled by the Ratio Map Manager), and makes this information available via a distributed hash table (DHT). For all data streams handled by SideStep, the Detour Group Manager actively searches for detour points that exhibit similar redirection dynamics. Once a candidate alternate path is found, the Race Manager splits the stream between the alternate and the current paths to evaluate their relative performance. The winner of this "race" is the path over which data is streamed.

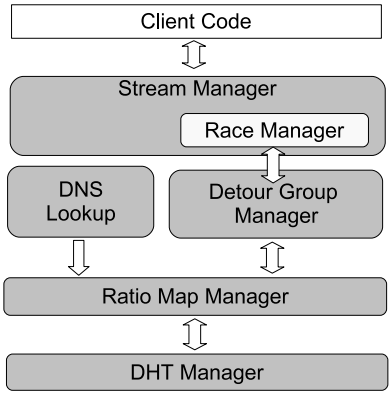


Figure 2: Architectural diagram for SideStep.

3.3 Managing Redirection Dynamics

We use CDN redirections dynamics as hints based on the hypothesis that if two nodes exhibit similar redirection behavior, they are likely to be along high-quality paths to one another and are thus good candidate detour points for each other [36]. In this section, we address the issue of how to encode this redirection behavior for each node, efficiently distribute these encodings and compare two nodes' redirection behavior.

A compact way to represent redirection behavior for each node is to use a map of ratios, where each ratio represents the frequency with which a node has been directed toward the corresponding replica server during the past time window [35]. 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:

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

More generally, the ratio map for a node a is a set of *(replica-server, ratio)* tuples represented as

$$\nu_a = \langle (r_k, f_k), (r_l, f_l), ..., (r_m, f_m) \rangle$$

For brevity, we use $\nu_{a,i}$ to represent the ratio of time f_i that node a is redirected to replica server r_i. Note that each node's ratio map contains only as many entries as replica servers seen by that node and that the sum of the f_i in any given ratio map equals one.

In the context of a detouring service, if two nodes have the same ratio map values, then they should be mapped to the same detour group. Similarly, if two nodes have completely different redirection behavior, they should be mapped to different detour groups. More generally, we would like a metric that, given two nodes, produces a continuum of values describing the similarity between the nodes' redirection behaviors. Based on our formulation of ratio maps, each node in our overlay can be represented as as a vertex in a general graph connected by edges labeled with the degree of overlap in their redirection frequency maps. Based on the premise that CDN redirections are primarily driven by network conditions, the structure of this graph can be used to arrange nodes in detour groups based on the *cosine similarity* of their ratio maps. Cosine similarity [29] is a mathematical measure how similar two vectors are, yielding values on a scale of $[0, 1]$. Treating a redirection map as a vector and given two hosts a and b, this can be formally defined as:

$$cos\_sim(a,b) = \frac{\sum_{i \in I_a} (\nu_{a,i} \times \nu_{b,i})}{\sqrt{\sum_{i \in I_a} \nu_{a,i}^2 \times \sum_{i \in I_b} \nu_{b,i}^2}}$$

Where I_a represents the set of replica servers to which node a has been redirected over the time window. Intuitively, the cosine similarity metric is analogous to taking the dot product of two vectors and normalizing the result. When the maps are identical, their resulting cosine similarity value is 1; when they are orthogonal (i.e., have no replica servers in common), the value is 0. Thus, to determine whether two nodes a and b are mapped to the same detour group, we can simply compute the cosine similarity of their redirection maps. 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.

Of course, before comparing two ratio maps, our system first must 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 mapped to a node identifier (e.g., a node's IP address) and each ratio-map entry is map between a replica server and the frequency with which it is witnessed. Given this structure, a DHT (which stores data as key-values pairs) is a natural and scalable solution for storing and retrieving such in-

formation. We discuss the details of SideStep's DHT-based technique for storing mapping data in Section 4.

As previously mentioned, 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 based in Asia and the South Pacific). 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 $\approx$ 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 3 demonstrates this property by comparing the RTT latency to a node in one detour group (the first hop of a detour path) to the end-to-end latency for the direct path to a node outside the group. The figure plots the cumulative distribution function (CDF) of two 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 customer Air Asia. For example, 50% of the detour points found using the Fox News customer name are at least 8 times closer to nodes inside the detour group than those that are outside. For the Air Asia customer name, however, only 33% of the detour group nodes provide the same level of proximity. 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.

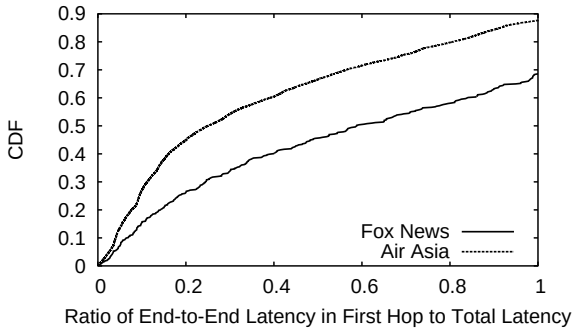


Figure 3: Ratio of first-hop latency to end-to-end latency over all CDN-recommended paths, demonstrating that different CDN customers lead to different detour-group characteristics.

While different CDN customer names form detour groups with diverse path characteristics, we show in Section 5.2 that this comes without significantly affecting detouring performance. Thus, to maintain scalability under heavy load, SideStep can spread detouring traffic over multiple detour groups by having nodes use different CDN customer names.

3.4 Validating Detour Paths

Once we have mapped 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 streaming data over each path concurrently, 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 until 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 destination reports to the source the throughput for the two paths, and the source uses the better one.

Although one can send arbitrary data over the detour path to measure its throughput, we choose to interleave client data over the two paths we are comparing. As a result, 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. We note that since evaluations of detour paths are typically short in duration, the overall impact on performance is negligible.

Our SideStep prototype improves end-to-end performance by selecting detour points that are in the same detour group as the source of data traffic. We expect to obtain similar results by locating nodes in the detour group for the destination of data traffic. It is, of course, possible to construct a two-hop overlay path where one hop is in the source node's detour group and the other is in that of the destination. Although it has been pointed that most of performance gains for detouring can be achieved using only one overlay hop [5], as part of our future work, we intend to investigate the potential benefits of two-hop detouring in the context of SideStep.

4 Implementation

This section describes the implementation of our SideStep prototype and how we ported an existing FTP client and

server code to use it. The current SideStep implementation is a user-level service written in Java for cross-platform portability and contains approximately 7,600 LOC. We discuss the implementation of each major SideStep components in turn.

4.1 Managing Redirection Dynamics

SideStep currently uses the Akamai and Limelight CDNs for hints to help drive a detouring system. 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 enjoys a global deployment of thousands of servers in hundreds of ISPs, we used the domain name associated with a video-on-demand site for a popular US television network. In this section, we detail how we efficiently manage redirections from these CDNs to determine candidate detour points for SideStep data transfers.

Computing Ratio Maps. As discussed in Section 3.3, we employ ratio maps to represent the mappings of nodes to replica servers, where each entry represents the frequency with which a node has been (re)directed toward the corresponding replica server. These mappings allow SideStep to form detour groups for selecting candidate one-hop overlay paths.

SideStep obtains ratio information by performing DNS lookups periodically. While we showed that the Akamai CDN refreshes replica-server mappings frequently enough as to be useful for network control [36], for this system we wanted to understand how *in*frequently these lookups can be performed without loss of accuracy in terms of replica-server mappings.

We analyzed ratio-map information generated by running SideStep on 133 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. To determine how ratio values change over time, we use the cosine similarity metric to compare the same node's ratio map at two different points in 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 4 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 0.1 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 self-similarity" values are 1. At the other extreme (on the same time scale), 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.

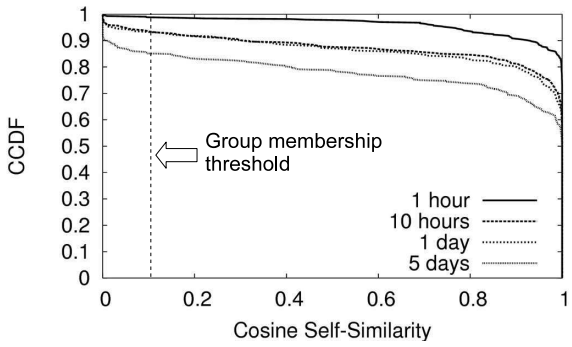


Figure 4: Cosine self-similarity over various time scales, indicating that DNS lookups can be performed as infrequently as once per hour.

In practice, SideStep performs ratio updates slightly more frequently than once per hour at different phases of a client's execution. When a node has no viable mapping information (i.e., during the bootstrapping phase), we increase the DNS lookup interval to once per minute. After the ratio value has stabilized, SideStep reduces the lookup interval to a parameterized default value set to one hour.

After each DNS translation, we recompute the ratio map and exponentially decay the existing values. The decay rate is set such that a value seen exactly once in a 24-hour period will be removed after a day has passed. SideStep periodically caches ratio mappings to local persistent storage. If SideStep must go offline, it can avoid the bootstrap period upon restarting by reusing the persistent mappings, provided that they were recorded recently.

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[1] 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.

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

Locating Detour Points. We use the cosine similarity metric on two nodes' ratio maps to determine whether they belong to the same detour group (Sec. 3.3). If cosine similarity is high, then the nodes are tightly bound to the same detour group.

SideStep currently supports storing and retrieving ratio maps from a service DHT (OpenDHT [28]) via a generic interface. After each ratio update, SideStep places the node's current ratio data in a distributed hash table (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.

As previously mentioned, a node's ratio map can change depending on the CDN customer associated with the DNS lookup. Thus, 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 from the same customer.

To retrieve ratio-map information for members of a detour group associated with a particular customer, a node follows the steps in Algorithm 1, which are described below. Recall that a ratio map entry $\nu_{a,i}$ corresponds to the percent of time a node a sees replica server i when performing a DNS lookup on a particular CDN customer (line 1). To search for detour points, a SideStep node first performs DHT lookups using each strongly mapped replica server cluster as a key (lines 3–4). From these lookups, it obtains a list of IP addresses for other nodes strongly mapped to those servers. Because *ratio maps* are required to determine group membership, SideStep performs a DHT lookup using each IP address as the key (lines 5–7), from which it obtains the ratio map, ν_p, for the corresponding node p. Once SideStep has obtained the ratio map for a node, it can compute the cosine similarity with its own ratio map (lines 8–9). If the result is greater than the threshold, t, the node is added to the list of potential detour nodes (line 10).

Scalability. 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.)

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

```
1  ν_a ⟵ local ratio map : ⟨ν_{a,1}, ν_{a,2}, ··· , ν_{a,m}⟩;
2  t ⟵ cosine similarity threshold;
3  foreach replica server cluster j : ν_{a,j} ≥ t do
4      peerAddresses ⟵ DHTLookup(j);
5      foreach p in peerAddresses do
6          /* get ratio map for peer p */
7          ν⃗_p ⟵ DHTLookup(p);
8          if CosineSimilarity(ν_p, ν_p) ≥ t then
9              AddToDetourGroup(p);
10         end
11     end
12 end
```

Algorithm 1: Replica-server mapping retrieval operation for a particular CDN customer. Ratio map entry $\nu_{a,i}$ corresponds to the percent of time a node a sees replica server i when performing a DNS lookup on the specified CDN customer.

Consequently, publishing mapping information requires at most $1 + c/t$ writes, where c is the number of CDN customers for which mapping information is maintained. 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 mappings for each CDN customer are orthogonal and there are $1/t$ entries for each 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 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. The total number of DHT operations performed to lookup all nodes in the same detour group scales linearly with the average number of nodes, n, mapped to each replica-server cluster. In the first step of a lookup, a node performs DHT reads only for replica-server clusters that it sees frequently (Algorithm 1, line 9), resulting in c/t DHT operations. This step returns a list of socket addresses of nodes in the same cluster. The system then performs a lookup for each socket address to retrieve the corresponding ratio map information. Thus, the maximum possible number of lookups performed is $n * c/t$, where n is the average number of socket addresses mapped to each replica-server cluster. Clearly, however, the maximum number of lookups that SideStep performs in practice ultimately depends on the duration of the associated file transfer.

4.2 Validating Detour Paths

SideStep uses hints about detour group membership to find good candidate detour points, but necessarily does not know the actual performance along the complete detour path. Consequently, SideStep must evaluate each detour path before determining whether to use the path preferentially over the direct one (or another detour path). We use *races* to compare the existing data-transfer path to a new one by splitting the data stream and sending it over each path concurrently.

As described in Section 3.4, during a race, 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.

Detour paths can outperform the direct path (or one another) for a variety of reasons, including lower latency, lower packet loss and higher available bandwidth. Due to the dynamics of such network conditions, the "best" path—be it a detour path or the direct path—may change during data transfer. Thus, SideStep performs races both periodically and dynamically in response to sudden changes in throughput along the current path.

SideStep provides a number of controls to ensure that races are performed efficiently. For instance, SideStep limits the frequency with which races are performed. Further, to ensure that our system does not iteratively probe paths that have relatively poor performance, a path that has lost a race cannot be re-evaluated until a certain amount of time has passed. Finally, if the system is currently using a detour path, it will re-evaluate the direct path periodically to ensure that detour paths continue to outperform the direct one.

For our SideStep experiments, we used a set of configurable parameters that work well in practice, which are presented in Table 1. An important part of our future work is to perform a detailed sensitivity study of how detouring performance changes in response different parameters.

4.3 Example Application: SideStep FTP Suite

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 for use with client applications. The API (see Fig. 5) has four basic calls: two for registering an incoming or outgoing data connection and two for requesting an input or output stream. (To close a SideStep connection, the client code simply closes the corresponding stream object.) The unique identifier (i.e., the `UniqueId` object returned by the API calls in Fig. 5) can optionally be used to set per-client preferences for finding detour group nodes, such as the cosine similarity threshold (t) and the CDN customer(s) to use for DNS lookup.

Parameter	Value
DNS lookup frequency	1 hour
Ratio map expiration	24 hours
Cosine similarity threshold, t	0.1
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 1: SideStep default parameters.

```
UniqueId createDataConnection(SocketAddress endpoint)
UniqueId listenForConnection(SocketAddress endpoint)
OutputStream getOutputStream(UniqueId transferId)
InputStream getInputStream(UniqueId transferId)
```

Figure 5: SideStep API.

To demonstrate the broad applicability of SideStep, we modified an existing open-source FTP client and server 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 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 to use SideStep requires changing only the source of the stream so that it is provided by SideStep instead of the default network library.

The SideStep FTP suite, DraFTP, is available publicly online[3] 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.

5 Evaluation

The goal of our evaluation is to examine SideStep performance over time and across variety of geographic regions. Recall that SideStep relies on CDN redirection dynamics as hints for making detouring decisions, and these dynamics respond to real-time changes in network conditions [36]. Logically then, the best approach to evaluate SideStep is through a widely-deployed, experimental testbed such as PlanetLab [26].

During a six-week period (late January to early March, 2007), we evaluated the effectiveness of our system in terms

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

of finding high quality detour paths between 6,336 distinct source–destination pairs over 133 PlanetLab nodes. While the reported results are, naturally, specific to our experimental testbed and the particular time of our experiments, we believe they indicate treds that are likely to continue in other SideStep deployments.

5.1 Experimental Approach

We conducted file-transfer experiments to characterize what type of source–destination pairs can benefit from CDN-based detouring. A simple way to explore this problem is to explore *all* distinct paths in our deployment; however, with 133 nodes, this approach requires far too much time and bandwidth.

Instead, for each round of experiments, we select source–destination pairs such that every source is likely to find at least one detour path during the experiment. This is based on the observation that if two nodes appear in the same detour group, they would have seen each other as candidate detour points for data traffic in the recent past and are likely to do so again in the near future. More importantly, if a node does not appear in the same detour group with any other node, it is *highly unlikely* to find any other node as a detour point in the near future. Thus, we first deploy SideStep on PlanetLab nodes without running experiments and record the ratio maps for each node. We use these ratio maps to calculate nodes' detour groups and determine the set of source–destination pairs to explore.

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 any detour group nodes were found) according to our description in the previous section. We used 30 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.

5.2 Cumulative Results

In this section, we present cumulative results from our experiments in which 9,506 paths were evaluated. Figure 6 shows a CDF of the observed performance benefits of CDN-based detouring, in terms of end-to-end throughput. The x-axis (log scale) represents the *ratio* of throughput over the direct path to the throughput over a detour path found by SideStep. The y-axis specifies the percent of values less than or equal to a given ratio. We separate the performance for each detour path according to its associated detour group, and label these groups using the CDN customer that we used to form them. Additionally, we providing a curve cumulating results over all detour groups.

The figure shows that, regardless of CDN customer name, over half (53%) of the alternative paths identified through CDN-based hints yielded performance improvements over the default one. Of those, 75% resulted in noticeable TCP throughput improvements (performance increasing by at least 10% over the direct path) while over 11% more than doubled the direct-path throughput. These results, which are comparable to those obtained using $O(n^2)$ measurement overhead, were obtained using a technique that incurred a *small, constant overhead* per node. Even when a detour path recommended by SideStep performs worse than the direct one, the former is never actually used as the primary path for data transfer. Besides the potential temporary delay incurred by racing data over two paths in parallel, our system incurs no overhead by validating these hints. We also note that the curves for different detour groups are very similar in shape. This suggests that one can use different CDN customer names to provide access to different detour groups without significantly affecting the quality of detouring performance.

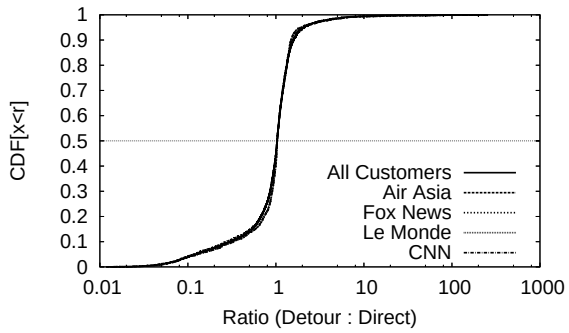


Figure 6: Ratio of detour throughput to direct-path throughput. Overall, more than half of the observed detour paths increase performance. Using different CDN customer names provides access to different detour groups without significantly affecting the quality of detouring performance.

We further analyzed this data to determine the kinds of performance gains witnessed on a per-path basis for the over 2,300 unique detour paths examined in our study. We found that 60% of the paths found offered up to a 10% performance improvement over the direct path, and 44% of the paths offered up to a 25% improvement in throughput. This further demonstrates that a CDN-based detouring service can achieve significant performance benefits over a large number of paths.

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 ones found by SideStep.

First, we use the `traceroute` measurements to compare end-to-end latency between the default Internet path and the one based on CDN hints. Figure 7 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.

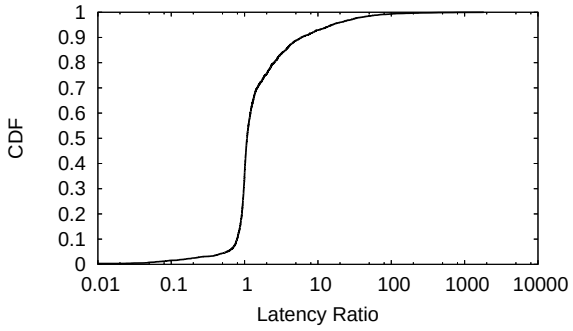


Figure 7: Ratio of direct-path to detour-path latencies, showing that nearly half of the detour paths significantly7 improve end-to-end latency.

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 8 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 finds diverse Internet paths; moreover, as Fig. 3 shows, the majority of these paths are along quality paths in terms of latency.

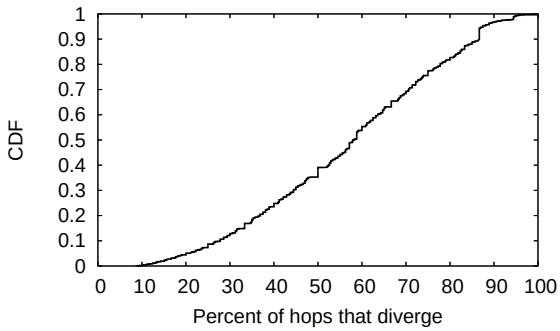


Figure 8: Divergence between direct and detour paths in terms of percent of total hops, demonstrating that SideStep finds detour paths offering significant path diversity.

Next, we use the `traceroute` data to compare loss rates along the alternate Internet paths. Figure 9 presents a CDF of the *difference* in loss rates between the direct path and the CDN-based alternate path, over all the experiments. Unlike the previous figures, we plot the difference instead of the ratio because loss rates exhibit a much smaller range of values than the other metrics. We compute these loss rates by determining the percentage of `traceroute` measurements dropped by routers during the entire measurement. They are *instantaneous* because they reflect the result of a small sample of values measured in response to an event and thus they are susceptible to certain biases. For example, if a race is conducted in response to a drop in throughput caused by congestion, then the loss rate reported by our system will likely be significantly higher than the *average* loss rate along that path during the entire experiment. Also, our loss rates are susceptible to bias due to routers dropping ICMP messages even when they are not dropping data packets.

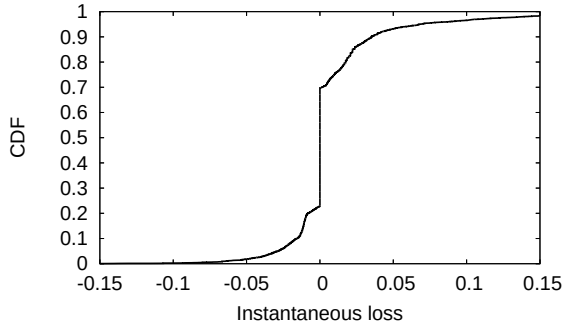


Figure 9: Difference in instantaneous loss rates between direct and detour path, indicating that a majority of detour paths exhibit loss rates that are as good or better than the direct ones.

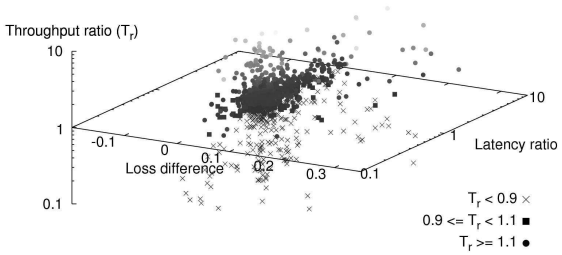
The figure clearly shows that loss rates along the detour path are as low or lower than the direct path more than 75% of the time. More importantly, the detour path reduces loss over the direct path more than 30% of the time. Thus, in the majority of cases, SideStep finds high-quality detour paths in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

Now we examine how throughput gains are related to the basic path characteristics observed via `traceroute`. Figure 10(a) presents a scatter plot using the result of all races conducted during our experiments. Each point (x, y, z) represents the following data for a single validation race: x is the difference in instantaneous loss between the detour path and the direct path; y is the ratio of the direct-path latency to the detour one; z is the ratio of detour-path throughput to the direct one (denoted T_r). It is immediately clear that most of the points are clustered near the zero point of the x-axis, where the difference in loss between the two paths is small, which Fig. 9 demonstrates is a common case.

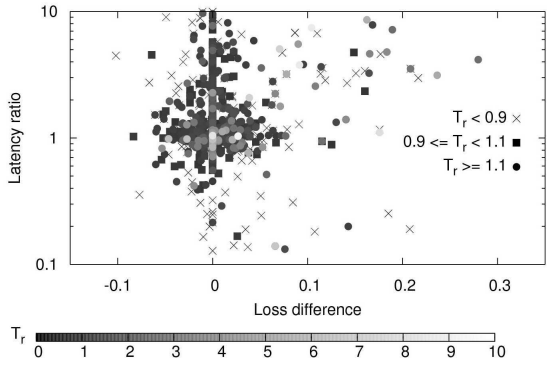
Figure 10(b) projects the data onto the XY plane to more clearly demonstrate any trends in the plot. The upper right quadrant represents lower latency and lower path loss along the detour path, while the bottom left quadrant represents a detour path with worse characteristics than the direct path. It is immediately clear that SideStep can see gains across a wide range of latency and loss values. It is also clear that

SideStep is unlikely to find performance gains along the detour path if its latency is greater than that of the direct path, as indicated by the relatively large fraction of X's below the ratio value of 1.

Finally, we observe that latency and loss path characteristics are, at best, weakly correlated with throughput improvements along a path. For example, even when latency and loss are low (the upper right quadrant of the graph), there is still a small number of X's indicating relatively low throughput. This occurs because factors such as available bandwidth and processing load at detour points can reverse any performance benefit from a higher quality path.



(a) Three-dimensional view.



(b) Projected onto the XY-plane

Figure 10: Scatter plot of the relationship among latency ratios, path loss differences and throughput ratios between the direct path and detour path.

Lastly, we look at how time-of-day effects the quality of detouring results. Figure 11 presents a CDF of throughput ratios (log scale) with each curve representing a different period of the day: early morning, late morning, afternoon and evening in the time zone of the source of the data transfer. We note that the curves have similar shapes, but there is clear separation among the curves at values significantly greater than or less than 1, indicating that time of day can affect detouring results. The reason that the effect is not more pronounced is that many detouring paths cross a large number of time zones, and our current implementation only optimizes the source half of the path. Thus, time-of-day effects near the destination can interfere with those at the source.

In Fig. 12, we plot only the values greater than 1 and

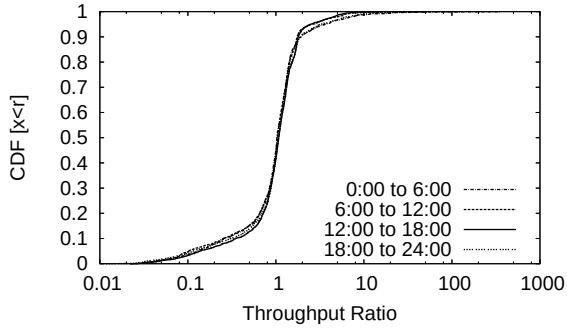


Figure 11: Time-of-day comparisons between detour and direct path throughput for all races, showing that throughput gains are sensitive to busy periods.

place the figure on a linear scale to better show the difference among the time-of-day curves. In this graph, it becomes clear that larger performance gains are seen in the off-peak evening hours. This indicates that off-peak hours provide our detouring service more available bandwidth for improving throughput along alternate paths.

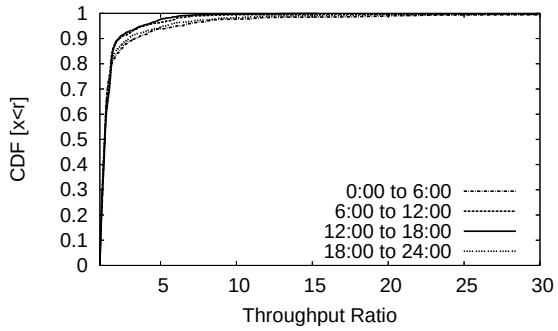
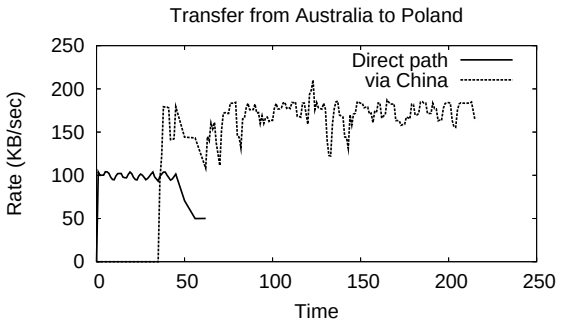


Figure 12: Time-of-day comparisons between detour and direct path throughput for detour paths that outperformed direct ones. The figure shows more bandwidth available along detour paths during off-peak hours.

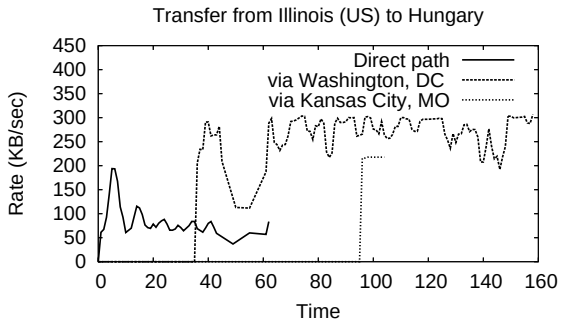
5.3 Detouring Examples

In this section, we detail the operation of our detouring service using graphs depicting examples of the throughput observed while transferring 30 MB files using SideStep across diverse geographic regions. Figure 13 contains four file transfers between four distinct pairs of nodes in our experimental testbed. In each subfigure, the y-axis represents end-to-end throughput and the x-axis represents time. Each curve indicates a different path between source and destination, and the solid curve always represents the direct path for a given pair of endpoints. We describe each transfer in turn.

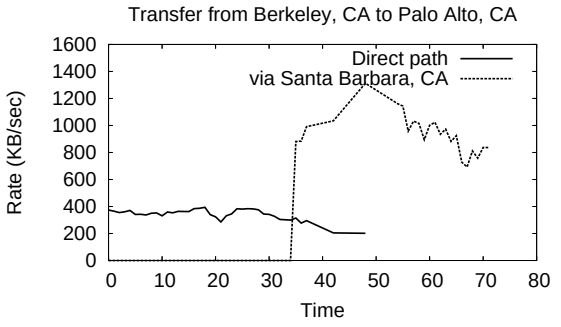
Figure 13(a) shows throughput for a transfer between a source node in Australia and a destination node in Poland. In this specific example, SideStep found that a node in China was in the same detour group as the source node



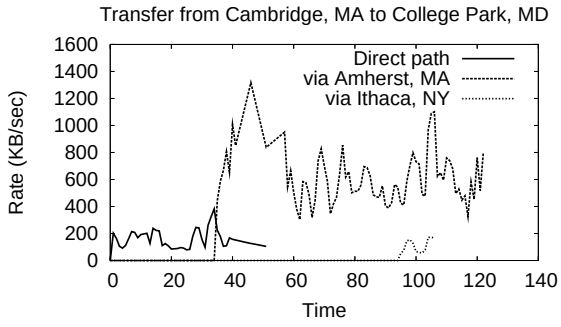
(a) Cross-continental path, **From:** Australia, **To:** Poland, **Via:** China



(b) Cross-continental path originating in the US, **From:** Illinois (US) **To:** Hungary **Via:** Washington, D.C. (US)



(c) Intrastate path, **From:** Berkeley, CA, **To:** Palo Alto, CA, **Via:** Santa Barbara, CA



(d) Interstate path, **From:** Cambridge, MA, **To:** College Park, MD, **Via:** Amherst, MA

Figure 13: Example traces of throughput during transfers using SideStep.

in Australia, so it evaluated the path by racing the alternate path through China with the direct path. In this case, the alternate path clearly wins the race, and SideStep streams data exclusively through China. This figure demonstrates that SideStep can find significant end-to-end improvements along public cross-continental paths outside the United States.

In Fig. 13(b), we demonstrate another cross-continental path, this one with the source node located in the Midwest region of the US and the destination node located in Hungary. SideStep first finds a one-hop path through a node located in Washington, D.C. on the East Coast of the US. After the racing period, it is clear that the detour path through Washington is better than the direct Internet path, so SideStep switches to the former. Unlike the previous figure, however, this one shows that SideStep finds another candidate detour point—this one in Kansas City, MO (located in the Midwestern US). In this case, the second detour path indeed outperforms the direct path measured previously, but it does not outperform the detour path through Washington, D.C. Thus, SideStep marks the Kansas City path as visited, and does not test it again until the refresh interval for the file transfer has expired.

Up to this point, we have demonstrated only cross-continental paths. It is important to note, however, that SideStep can improve performance for shorter range transfers. Figure 13(c) shows the performance of a file transfer between a source node in Berkeley, CA and a destination node in Palo Alto, CA—both of which are in northern California. Again, SideStep locates a detour node in the same detour group of the source, specifically in Santa Barbara, CA. The resulting throughput burst rate more than triples the direct-path throughput, and it eventually stabilizes at a rate that is more than double that of the direct path.

Finally, we present another example of performance gains over relatively short paths (Fig. 13(d)). In this case, the source node is located in Cambridge, MA and the destination node is located in College Park, Maryland, both of which are on the East Coast of the US. SideStep first locates a detour path through Amherst, MA that (despite high variability in throughput) significantly improves performance over the direct path. Later during the transfer, SideStep discovers a detour point in Ithaca, NY; however, it clearly does not outperform the path through Amherst, nor does it appear to outperform the previously measured direct path. This path is quickly abandoned.

This section demonstrated SideStep's successful operation in a variety of transfer scenarios. Note that in none of the above examples did overall end-to-end throughput between the source and destination drop during hint validation, even when the hints proved to be wrong.

6 Discussion and Future Work

This work presents results from a detouring service, SideStep, that relies on CDN redirection dynamics as hints to achieve high scalability. In this section we detail the limits of our approach, discuss issues that must be addressed should SideStep 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 that take longer than one minute on average (in our experiments, this corresponds to files on the order of 10s of MBs or more). The minimum file size that can see improved performance is limited by the latency for finding and validating detour paths, relative to the total time required to download the file. If the entire file can be transferred along the direct path in less time than is required to find a good alternate path, then SideStep cannot provide any benefit to the file transfer. While SideStep could support relatively small files by performing races more frequently than in its current implementation, we do not expect significant performance improvements for arbitrarily small flows.

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. We believe that a similar validation technique will allow SideStep to evaluate the quality of alternate paths; however, there are many open issues regarding the frequency with which to evaluate these paths and the extent to which delays at the middle hop can reduce the raw performance gain along those paths.

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 decline to accept 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 total available bandwidth to the SideStep system, we expect that a credit-based mechanism (e.g., [14]) 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 short 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 and 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.

7 Conclusions

This paper demonstrated that one can build a high-performance detouring service by reusing available network information, gathered at low cost from existing oracles, that would otherwise be forced to probe their environment independently. In particular, we showed how SideStep uses CDN redirections dynamics, seen as hints regarding

good candidate detour points, to locate good paths among these alternatives without additional monitoring. We built a service to support CDN-based, fe-hop source routing and implemented it as a ready-to-use service that is simple to integrate into existing applications. Our extensive measurements of a wide-area deployment in PlanetLab demonstrate the feasibility and benefits of this novel approach to detouring.

In summary, we found that in more than half of our experiments, our service can exploit CDN redirection dynamics to discover paths that have end-to-end *throughput* larger than that of the direct path. For those that do not improve performance, we developed efficient techniques for evaluating the validity of CDN-based hints. We presented a `traceroute`-based study showing that the majority of CDN-based paths exhibited better basic path characteristics than the default Internet paths. Finally, we deployed SideStep on a large number of PlanetLab nodes and provide access to the service via a portable, publicly available library.

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.

[3] AKAMAI. Sureroute, May 2003.

[4] AKAMAI. Akamai introduces first-of-its-kind, real-time view into health of the Internet, June 2007.

[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.

[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] 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).

[10] 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).

[11] 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).

[12] 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).

[13] 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).

[14] GUPTA, M., JUDGE, P., AND AMMAR, M. A reputation system for peer-to-peer networks. In *Proc. of NOSSDAV* (2003), ACM Press, pp. 144–152.

[15] KANGASHARJU, J., ROSS, K., AND ROBERTS, J. Performance evaluation of redirection schemes in content distribution networks. *Computer Communications 24*, 2 (2001), 207–214.

[16] LABOVITZ, C., MALAN, G. R., AND JAHANIAN, F. Internet routing instability. In *Proc. of ACM SIGCOMM* (September 1997).

[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] 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).

[20] MIRROR IMAGE. Mirror image CDN.

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

[22] NAKAO, A., PETERSON, L., AND BAVIER, A. A routing underlay for overlay networks. In *Proc. of ACM SIGCOMM* (August 2003).

[23] NETWORKS, L. Limelight networks CDN.

[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.

[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] SALTON, G., AND MCGILL, M. J. *Introduction to modern information retrieval*. McGraw-Hill, New York, NY, 1986.

[30] SARKAR, P., AND HARTMAN, J. Efficient cooperative caching using hints. In *Proc. of USENIX OSDI* (October 1996).

[31] 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.

[32] 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*.

[33] SESHAN, S., STEMM, M., AND KATZ, R. H. SPAND: shared passive network performance discovery. In *Proc. of USENIX USITS* (December 1997).

[34] SHAIKH, A., TEWARI, R., AND AGRAWAL, M. On the effectiveness of DNS-based server selection. In *Proc. of IEEE INFOCOM* (April 2001).

[35] SU, A.-J., CHOFFNES, D., BUSTAMANTE, F. E., AND KUZMANOVIC, A. Relative network positioning via cdn redirections. Tech. Report NWU-EECS-2007-03, Northwestern University, January 2007.

[36] 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).

[37] TANG, C., AND MCKINLEY, P. A distributed multipath computation framework for overlay network applications. Tech. rep., Michigan State University, 2004.

[38] WAWRZONIAK, M., PETERSON, L., AND ROSCOE, T. Sophia: An information plane for networked systems. In *Proc. of HotNets* (November 2003).

[39] 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).