Moving Beyond End-to-End Path Information to Optimize CDN Performance

Rupa Krishnan[§]   Harsha V. Madhyastha[‡]   Sridhar Srinivasan[§]   Sushant Jain[§]

Arvind Krishnamurthy[£]   Thomas Anderson[£]   Jie Gao[†]

[§] Google Inc. [‡] University of California San Diego   [£] University of Washington [†] Stony Brook University

ABSTRACT

Replicating content across a geographically distributed set of servers and redirecting clients to the closest server in terms of latency has emerged as a common paradigm for improving client performance. In this paper, we analyze latencies measured from servers in Google's content distribution network (CDN) to clients all across the Internet to study the effectiveness of latency-based server selection. Our main result is that redirecting every client to the server with least latency does not suffice to optimize client latencies. First, even though most clients are served by a geographically nearby CDN node, a sizeable fraction of clients experience latencies several tens of milliseconds higher than other clients in the same region. Second, we find that queueing delays often override the benefits of a client interacting with a nearby server.

To help the administrators of Google's CDN cope with these problems, we have built a system called *WhyHigh*. First, *WhyHigh* measures client latencies across all nodes in the CDN and correlates measurements to identify the prefixes affected by inflated latencies. Second, since clients in several thousand prefixes have poor latencies, *WhyHigh* prioritizes problems based on the impact that solving them would have, e.g., by identifying either an AS path common to several inflated prefixes or a CDN node where path inflation is widespread. Finally, *WhyHigh* diagnoses the causes for inflated latencies using active measurements such as traceroutes and pings, in combination with datasets such as BGP paths and flow records. Typical causes discovered include lack of peering, routing misconfigurations, and side-effects of traffic engineering. We have used *WhyHigh* to diagnose several instances of inflated latencies, and our efforts over the course of a year have significantly helped improve the performance offered to clients by Google's CDN.

Categories and Subject Descriptors

C.2.3 [**Computer Communication Networks**]: Network Operations, Network Management

General Terms

Algorithms, Measurements, Management, Performance

Keywords

Network Measurements, Content Distribution Networks

1. INTRODUCTION

The use of content distribution networks (CDNs) has emerged as a popular technique to improve client performance in client-server applications. Instead of hosting all content on a server (or a cluster of servers) at a single location, content providers today replicate their content across a collection of geographically distributed servers or nodes. Different clients are redirected to different servers both for load balancing and to reduce client-perceived response times. The most common redirection method is based on latency, i.e., each client is redirected to the server to which it has the lowest latency in order to reduce the download time to fetch the hosted content. For example, several CDNs that run on PlanetLab use OASIS [7] to redirect clients to the node with least estimated latency.

Google's CDN similarly employs latency-based redirection. Common knowledge to optimize client latencies in such a setting is to establish more CDN nodes [9, 19] so as to have a CDN node in the proximity of every client. However, we found that this does not suffice. For example, to reduce the latencies for clients in Japan when fetching content from Google, a new node in Japan was added to the Google CDN. While this decreased the minimum round trip times (RTTs) for clients in that region, the worst-case RTTs remained unchanged. This case showed that rather than investing in setting up new CDN nodes, we need to first understand whether the existing nodes yield the best performance possible.

In this paper, we use measurements gathered at Google's nodes to study the effectiveness of latency-based redirection in optimizing CDN performance. We analyze RTTs measured to clients spanning approximately 170K prefixes spread across the world by monitoring the traffic Google exchanges with these clients. Our analysis of this data shows that even though latency-based redirection results in most clients being served by a geographically proximate node, the improvement in client-perceived RTTs falls short of what one would expect in return for the investment in setting up the CDN. We find two primary causes for poor latencies.

First, we find that geographically nearby clients often see widely different latencies even though they are served by the same CDN node. In our data, we find that clients in more than 20% of prefixes are served by paths on which RTTs are 50ms more than that observed to other clients in the same geographic region. Such discrepancies in latencies are because some clients have circuitous routes to or back from the CDN node serving them. To detect and diagnose these cases, we move beyond using end-to-end latency information and delve into the Internet's routing to optimize client performance. Second, we observe that connections to most clients are impacted by significant queueing delays. In this paper,

we characterize the extent of such queueing related overhead on client RTTs.

Given these problems, it is important that administrators be able to quantify the performance gains from the significant expense and effort that goes into deploying nodes in a CDN. Based on the analysis of our dataset and discussions with the network administrators, we have built a system called *WhyHigh* to aid administrators in this task.

WhyHigh first identifies clients with poor latencies as ones which experience latencies much higher than that along the best path to the same region. In a CDN that has several distributed nodes with each node serving clients in thousands of prefixes, identifying clients with poor performance is itself a significant challenge. Since clients in thousands of prefixes are determined to have inflated latencies, an administrator cannot investigate all of these simultaneously. To help the admins prioritize their efforts at troubleshooting, *WhyHigh* attempts to group together all prefixes likely affected by the same underlying cause. For example, we find that when a prefix suffers from inflated latencies, other prefixes served by the same AS path are also likely affected. *WhyHigh* also computes metrics that help compare different nodes in the CDN so that problems at the worst nodes can be investigated first to have higher impact.

WhyHigh then attempts to pinpoint the causes for the instances of latency inflation seen at each node. To infer these causes, *WhyHigh* combines several different data sources such as BGP tables from routers, mapping of routers to geographic locations, RTT logs for connections from clients, and traffic volume information. In addition, the system also performs active probes such as traceroutes and pings when necessary. Typical causes inferred by *WhyHigh* include lack of peering, routing misconfigurations, and side-effects of traffic engineering.

We have employed *WhyHigh* over the course of almost a year in troubleshooting latency inflation observed on paths between Google's nodes and clients. This has helped us identify and resolve several instances of inefficient routing, resulting in reduced latencies to a large number of prefixes. The metrics of CDN performance provided by *WhyHigh* have also allowed network administrators to monitor the results of their efforts by providing a global picture of how different nodes in the CDN are performing relative to each other.

The organization of the rest of the paper is as follows. Section 2 provides an overview of Google's CDN architecture, the dataset gathered from the CDN that we use in this paper, and our goals in analyzing this dataset. Section 3 describes the poor latencies obtained with latency-based redirection and demonstrates that routing inefficiencies and queueing delays are the primary limiting factors. In Section 4, we describe the *WhyHigh* system that we developed to diagnose instances of latency inflation. We then present a few representative case studies of troubleshooting inflated paths and summarize the impact of using *WhyHigh* in Section 5. In Section 6, we draw from our experiences and discuss some limitations of *Why-High*. We conclude with a look at related and future work in Sections 7 and 8.

2. OVERVIEW

In this section, we first provide a high-level overview of the architecture of Google's CDN deployment. Next, we present the goals of our work, while also clarifying what are non-goals. Finally, we describe the RTT dataset we gathered from Google's servers and present the preprocessing we performed on the data before using it for our analysis in the rest of the paper.
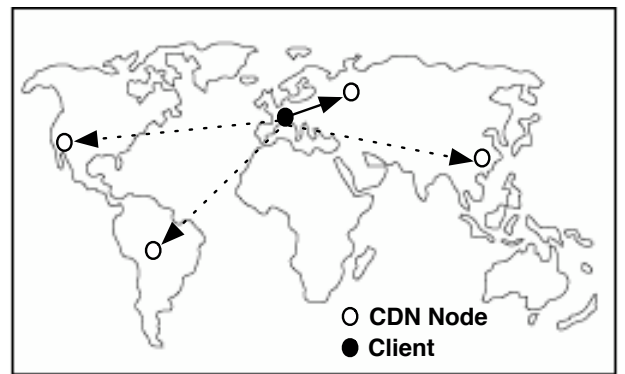
2.1 CDN Architecture

Figure 1: **Architecture of Google CDN. Dotted lines indicate paths used primarily for measuring RTT. Bold line indicates path on which traffic is primarily served.**

Figure 1 presents a simplified view of the architecture of the content distribution network operated by Google. Google's CDN comprises several nodes that are spread across the globe. To reduce the response times perceived by clients in fetching Google's content, the CDN aims to redirect each client to the node to which it has the least latency. For this, all clients in the same routable prefix are grouped together. Grouping of clients into prefixes is naturally performed as part of Internet routing, and the current set of routable prefixes are discovered from BGP updates received at Google.

To gather a map of latencies from CDN nodes to prefixes, every so often, a client is redirected to a random node and the RTT along the path between the client and the node is measured by passively monitoring the TCP transfer at the node. Since routing in the Internet is largely identical to all addresses within a prefix, the RTT measured to a client is taken to be representative of the client's prefix. Such RTT measurements gathered over time, and refreshed periodically, are used to determine the CDN node that is closest in terms of latency to each prefix. Thereafter, whenever a client attempts to fetch content hosted on the CDN, the client is redirected to the node determined to have the least latency to its prefix. This redirection however is based on the prefix corresponding to the IP address of the DNS nameserver that resolves the URL of the content on the client's behalf, which is typically co-located with the client.

2.2 Goals

In analyzing the performance offered by the CDN architecture described above, we have three primary goals.

- Understand the efficacy of latency-based redirection in enabling a CDN to deliver the best RTTs possible to its clients.
- Identify the broad categories of causes for poor RTTs experienced by clients.
- Implement a system to detect instances of poor RTTs and diagnose the root causes underlying them.

2.3 Non-goals

Apart from latency, TCP transfer times can also be dictated by path properties such as loss rate and bandwidth capacity. However, in this paper, we restrict our focus to optimizing end-to-end path latency, which is the dominant factor for short TCP transfers [3]. Further, our focus is on the component of end-to-end latency incurred through the network. In practice, response times perceived by clients can also depend on application-level overheads, e.g., delays incurred in rendering the content in a Web browser.

Latency-based redirection as described in the CDN architecture above can be rendered ineffective in a couple of scenarios. Clients which use a distant DNS nameserver may be redirected to a distant node, since the CDN will redirect the client to a node close to the nameserver. Redirection decisions made at the granularity of prefixes can be sub-optimal for prefixes in which clients are significantly geographically distributed. In this paper, we focus on causes for poor client latencies even when the client's prefix is localized and the client is co-located with its nameserver.

2.4 Measurement Dataset

The dataset we use for most of the analysis in this paper consists of RTT data measured by all of Google's nodes to the 170K network prefixes they serve over the period of one day. The median number of RTT samples per prefix is approximately 2000. For each TCP connection at a node, the node logs the time between the exchange of the SYN-ACK and SYN-ACK-ACK packets during connection setup. The RTT thus measured includes only the propagation and queueing delays on the forward and reverse paths traversed by the connection; transmission delay dependent on the client's access link bandwidth is not a factor because typically SYN-ACK and SYN-ACK-ACK packets are small. Though it is possible for the size of a SYN-ACK-ACK packet to be inflated by having a HTTP request piggy-backed on it, we found less than 1% of connections with such piggy-backed packets in a sample tcpdump.

Our analysis is based on RTT logs, BGP tables, and Netflow records obtained on 2 days—one in May 2008 and one in August 2008. We first use data from May to demonstrate the problems that clients face even after they have been redirected to the closest node in terms of latency. We later use data from August in Section 5 to illustrate the performance gains achieved through our efforts in troubleshooting poor client latencies.

2.4.1 Data Pre-Processing

We perform a couple stages of processing on our measurements before using them for analysis. First, we use BGP snapshots from different routers within the Google network to map every client to the routable prefix to which it belongs. Second, we tag prefixes with geographic information obtained from a commercial geolocation database. The location of each prefix was determined at the granularity of a region, which roughly corresponds to a province within a country. We applied three stages of processing to prune out prefixes with incorrect geographical information from our dataset.

First, we identify and eliminate any prefix such that the minimum RTT measured to it is impossible based on its estimated geographic location, i.e., RTT is much lesser than the time taken if a straight cable were laid between the node and the prefix. Prior work [10] on the geolocation of Internet hosts observed that the expected RTT through the Internet is that obtained when bits travel at $\frac{4}{9}^{th}$ the speed of light in vacuum. We refer to the RTT computed based on this assumption to be the *Geo-RTT* of a prefix. We eliminated all prefixes whose minimum RTT to any of the Google nodes was significantly lower than the Geo-RTT estimate to that node. This step prunes out 21K of the 173K prefixes in our dataset.

Second, we use the *"confidence"* information tagged along with data obtained from our geolocation database to discard information that is probably incorrect. Our geolocation product returns three attributes for every location—one each for country, region, and city. Each of these three attributes takes an integer value from 0 to 5, indicating the probability of the corresponding attribute being correct. The 0-to-5 scale equally divides up the confidence scale; a value of 5 indicates that the probability of the location being correct varies from 83% to 100%, a value of 4 for the range 67% to
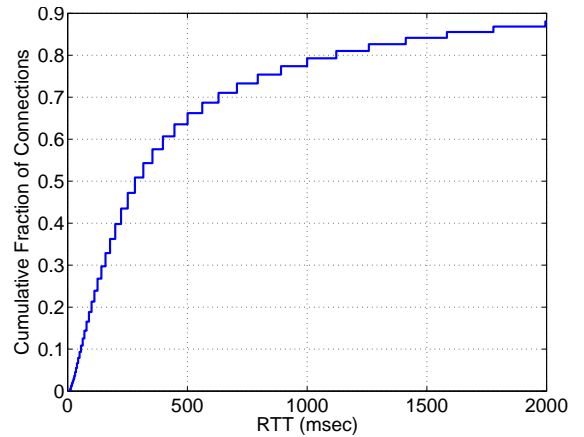

Figure 2: Distribution of measured RTTs.

83%, and so on. Since all of our analysis in this paper uses geolocation information at the granularity of region, we discard all prefixes whose location has a country confidence less than 5 and a region confidence less than 4. This prunes out a further 42K prefixes. In our sample of IP addresses with known locations used for validation, we find the 90th percentile error in location to be approximately 200 miles when restricting ourselves to locations associated with a country confidence of 5 and a region confidence greater than or equal to 4.

Third, some prefixes may span a large geographical region. For such prefixes, variation in RTT across clients could be because of their varying distance to the Google node serving that prefix. Since our focus in this paper is on studying poor RTTs even when prefixes are geographically localized, we eliminate these distributed prefixes from consideration. To identify such prefixes, we compute the width of each prefix in the dataset by choosing a few hundred IP addresses at random from the prefix, determining the locations for these addresses, and computing the maximum distance between pairs of chosen addresses. We discard all prefixes with a width greater than 100 miles. This eliminates an additional 5K prefixes.

3. UNDERSTANDING PATH LATENCY

In this section, we analyze our dataset to understand the RTTs seen by clients of Google's CDN infrastructure. First, we study the efficacy of latency-based redirection. Then, we investigate each potential cause for poor performance.

The motivation for this paper lies in the poor latencies seen by clients. Figure 2 plots the distribution of RTTs measured across all connections in our dataset. Even though each client is served by the CDN node measured to have the lowest latency to the client's prefix, RTTs greater than 400ms are measured on 40% of the connections served. We would expect to see significantly lower RTTs given the spread of Google's CDN; roughly 75% of prefixes have a node within 1000 miles (Geo-RTT less than 20ms) and roughly 50% of prefixes have a node within 500 miles (Geo-RTT less than 10ms). Moreover, 400ms is more than the typical RTT obtained on a path all the way around the globe, e.g., the RTT between Planet-Lab nodes on the US west coast and in India is roughly 300ms. This shows that in spite of replicating content across several CDN nodes to ensure that every client has a server nearby, which requires significant investment in infrastructure, the latencies experienced by clients are poor.

To understand the cause for these high RTTs, we break down each measured connection RTT into its components. RTTs mea-
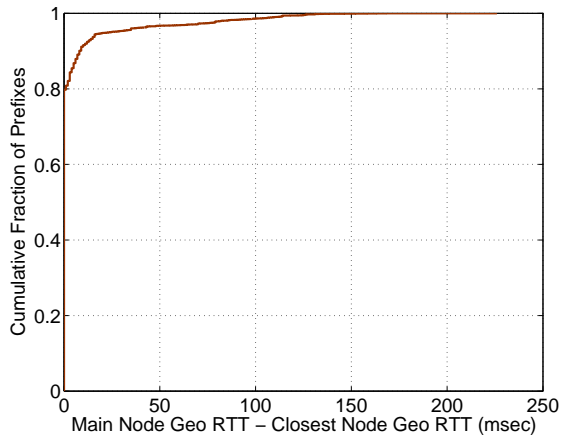
Figure 3: CDF across prefixes of the difference between Geo-RTTs from a prefix to the main node serving it and to its closest node. Most prefixes are served by geographically nearby nodes.

Figure 4: Distribution of the inflation in latency observed in the minimum and median RTT to a prefix as compared to the minimum RTT to any prefix in the same region. Inflation seen in a prefix's minimum RTT is predominantly due to routing inefficiencies, and additional inflation seen in the median is due to packet queueing.

Figure 5: Latency inflation w.r.t other prefixes in the region for (i) prefixes redirected to the closest node, and (ii) prefixes served by nodes farther away.

sured at the TCP layer have three main components—transmission delay (time to put a packet on to the wire), propagation delay (time spent from one end of the wire to the other end), and queueing delay (time spent by a packet waiting to be forwarded). Since our measurements are based on control packets typically of size 50 bytes, transmission delay is less than 1ms even on a dialup link. Therefore, a high RTT is the result of inflation in either propagation delay or queueing delay. Inflation in propagation delay could be the outcome of two factors. On the one hand, it could be that even though clients are redirected to the node with lowest latency, that node is typically far away from the client. Or, on the other hand, clients could have high RTTs even to the nearby node to which they are redirected. Beyond propagation delay, high RTTs could also be the result of packets getting queued up somewhere along the path between the clients and the nodes serving them. We next examine each of these potential causes for poor latencies in detail.

3.1 Effectiveness of Client Redirection

We first evaluate whether the cause for high RTTs is because clients are being redirected to nodes distant from them. For every prefix, we identify the main node serving it, i.e., the node from which most of the connections to this prefix were observed. We also identify for every prefix, the node geographically closest to it. Based on the CDN architecture described previously, the main node for each prefix should be the node to which it has the lowest RTT. However, there are a few cases where this property is violated. Clients can be redirected elsewhere if the lowest RTT node is under high load, the nameserver used by the client is not co-located with it, or network conditions change between when the RTT to the client's prefix was measured and when that information is used to make redirection decisions. Since our goal is to evaluate the performance of the CDN when clients are served by the node to which they have the least RTT, we drop all prefixes from our dataset where that property is not true. This prunes out an additional 14K prefixes.

Figure 3 plots the difference between the Geo-RTT, as previously defined, from a prefix to its main node and the Geo-RTT from that prefix to its geographically closest node. Clients in 80% of prefixes are served by their geographically closest node. Further, 92% of the prefixes are redirected to a node that is within 10ms of Geo-RTT from their closest node, which corresponds to a distance of 400 miles. This shows that latency-based redirection does result in most prefixes being served by nearby nodes, albeit we show later
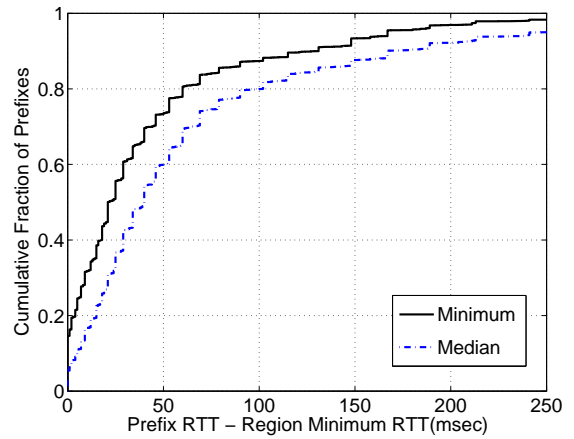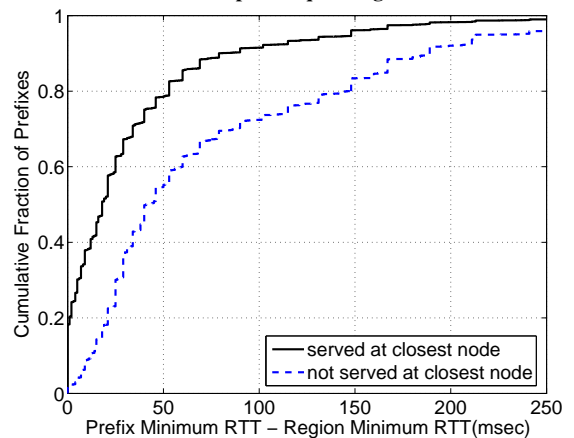
in Section 4 that this is not true for new nodes added to the CDN.

3.2 Characterizing Latency Inflation

We next investigate large propagation delay to nearby nodes as the potential cause for the high measured RTTs. For this, we compare the minimum RTT seen across connections originating at a prefix with the minimum RTT measured across all prefixes in the same region, both measured at the prefix's main node; given the large number of RTT samples in each prefix, the minimum RTT to a prefix is unlikely to have a queueing delay component. The solid line in Figure 4 makes this comparison. We see that more than 20% of prefixes experience minimum RTTs that are more than 50ms greater than the minimum RTT measured to other prefixes in the same region. This problem is even worse when considering prefixes served by new nodes, i.e., nodes active for less than 1 year; 45% of prefixes served at new nodes have minimum RTTs that are 50ms greater than other RTTs measured to the region (not shown in figure). This shows that although most prefixes are served by a node that is geographically close, and there exist good paths to the regions of these prefixes, routing inefficiencies result in inflated RTTs. Though our methodology will miss instances of inflation
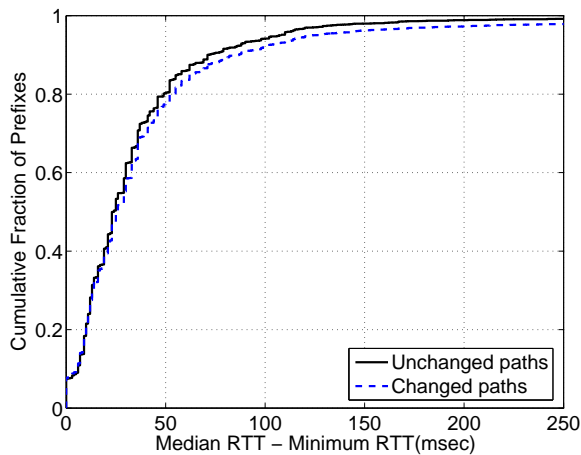
Figure 6: Comparing latency inflation between sets of prefixes partitioned based on whether the path to them changes across consecutive days.

Figure 7: Fraction of clients affected by significant queueing delays in inflated prefixes. In most inflated prefixes, most clients experience high latency overhead owing to queueing.

when all prefixes in a region are affected, we find that cases where the best RTT to a region is poor are rare.

These numbers show that the problem of inflated latencies due to inefficient routing is significantly worse than previously considered. Prior work [17] that looked at inflation between pairs of cities had found less than 5% of paths to have an inflation greater than 25ms. Our analysis of a much more extensive RTT dataset measured to several tens of thousands of prefixes across the Internet shows that more than 20% of paths have an inflation greater than 50ms, even though most paths are between clients and nearby nodes.

To further examine the inflation observed in the minimum RTT to a prefix, we partition prefixes in our dataset into two sets—1) prefixes that are redirected to the node geographically closest to them, and 2) all other prefixes (which are not redirected to their geographically closest node). Figure 3 shows that 80% of prefixes belong to the first partition and 20% belong to the second.

A prefix would be redirected to a node other than the geographically closest one only when there exists another node with lower latency to the prefix. Therefore, one would expect prefixes in the first set to have lower inflation, on average. To examine this hypothesis, Figure 5 plots the inflation observed in the minimum RTTs for prefixes in both sets. We see that even prefixes redirected to their closest node suffer from significant latency inflation—20% of prefixes have inflation greater than 50ms—and this is even more pronounced for prefixes redirected to farther servers. When considering only the prefixes served by new nodes (not shown in figure), this problem is even worse—in either set, 45% of prefixes suffer greater than 50ms inflation.

3.3 Characterizing Queueing Delays

Next, we analyze the other potential cause for high RTTs—packet queueing. Going back to Figure 4, we plot the inflation observed in the median RTT measured to a prefix compared to the minimum RTT to its region. We see that the additional latency overhead observed is significant; 40% of the prefixes incur an inflation greater than 50ms when considering the median. The variance in RTTs across different clients in a prefix because of geographic diversity is limited by the fact that our dataset only contains prefixes whose geographic width is at most 100 miles, which accounts for less than 5ms of RTT in the worst case. Also, variance in access link bandwidths across clients in a prefix cannot explain a difference of tens
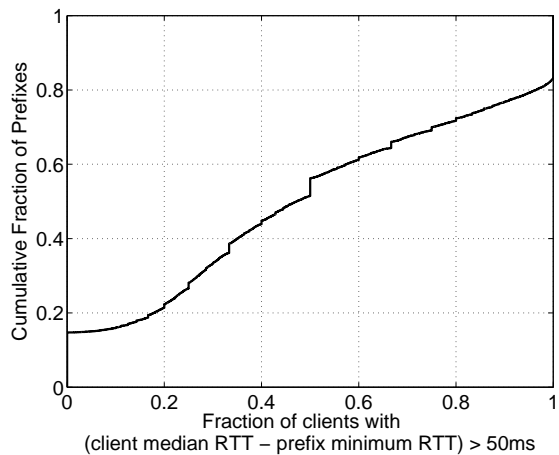
of milliseconds as our measurements are made over TCP control packets whose sizes are typically in the order of 50 bytes.

Therefore, the significant variance within RTTs measured across connections to the same prefix can be due to two reasons—1) the route between the prefix and its main node could vary over the course of a day, which is the duration of our dataset, or 2) packets could be getting queued along the route. To investigate the former, we ranked the prefixes based on the overhead measured in the median RTT to that prefix as compared to the minimum to the prefix. We considered the top 10K prefixes based on this ranking and issued traceroutes to them from their main nodes over the course of a day. We then re-issued these traceroutes the next day and partitioned the prefixes based on whether the routes measured were identical across the two days. To determine whether the two routes to a prefix across successive days were identical, we first performed the comparison at the granularity of router interfaces as measured by traceroute. If deemed non-identical, we mapped the routes to the granularity of Points-of-Presence (PoPs) using data from iPlane [12] and compared the PoP-level routes. 4K prefixes had routes unchanged across the two days whereas 6K prefixes had routes that we deemed to be non-identical.

Figure 6 plots the overhead seen in comparing the median and minimum RTTs to a prefix for prefixes in the two partitions. The distribution of latency overhead is pretty much identical irrespective of whether the route to the prefix is stationary or not. In this experiment, our determination of whether routing to a prefix is stationary was based only on the routes we measured from the CDN nodes to these prefixes and measured once per day. Therefore, we might have missed out on changes in routing on the path back from the prefix to the node or at finer timescales. However, the fact that accounting for even the limited degree of stationarity that we detect does not make a difference implies that queueing of packets, and not flux in routing, is the predominant cause for the huge variance observed in RTTs within a prefix. While our data does not enable us to determine whether packets are being queued in routers, in access links, or within the operating system of clients, large queueing delays have previously been measured [4] on the access links of end-hosts.

To determine if packet queueing is restricted to a few clients within each prefix and not to others, we conducted the following experiment. We considered the top 25K prefixes in the ranking computed above based on the difference between the median and

minimum RTTs to the prefix. In each of these prefixes, we computed for every client the difference between the median RTT measured to the client and the minimum RTT to the prefix, i.e., the median RTT inflation experienced by the client due to queueing. Figure 7 plots the fraction of clients in each prefix for which this difference is greater than 50ms. We see that in roughly half of the prefixes, more than half the clients have an overhead greater than 50ms in the median RTT as compared to the minimum to the same prefix. This shows that latency overheads caused by packet queueing are widespread, not restricted to a few clients within each prefix.

3.4 Summary

The results in this section demonstrate that although redirection of clients is essential to reduce client latency, that alone does not suffice. Redirection based on end-to-end RTTs results in most clients being served from a geographically nearby node. However, two factors result in significant latency inflation. First, a significant fraction of prefixes have inefficient routes to their nearby nodes, a problem which is even more pronounced when considering prefixes served by CDN nodes active for less than a year. Second, clients in most prefixes incur significant latency overheads due to queueing of packets. Isolating the cause or fixing the overheads caused by packet queueing is beyond the scope of this paper. We focus on diagnosing and fixing routing inefficiencies that lead to poor client RTTs.

4. *WhyHigh*: A SYSTEM FOR DIAGNOSING LATENCY INFLATION

In this section, we describe the design and implementation of *WhyHigh*, a system that we have built to detect and diagnose cases of inefficient routing from nodes in Google's CDN to clients. We follow up in the next section with a few representative cases of path inflation diagnosed by *WhyHigh* and provide an overview of the progress we have made in our efforts towards fixing the problems presented in Section 3.

In building *WhyHigh* to diagnose poor RTTs experienced by clients, we follow a three-staged approach—1) *identify* prefixes affected by inefficient routing, 2) *prioritize* these prefixes for investigation by administrators, and 3) *diagnose* causes for the identified problems. We now describe the techniques used by *WhyHigh* in each of these subcomponents in detail.

4.1 Identifying Inflated Prefixes

At each CDN node, we measure the RTT over every connection. We then identify prefixes with inflated RTTs, which we refer to as *inflated prefixes*. To identify inflated prefixes at a node, we compare the minimum RTT measured at the node across all connections to the prefix with the minimum RTT measured at the same node across all connections to clients within the prefix's region. We declare a prefix to be inflated if this difference is greater than 50ms. As we argued previously in Section 3, inflation in latency observed in the minimum RTT to a prefix is because of inefficient routing. Thus, we use all inflated prefixes identified using this process as candidates for further troubleshooting.

4.2 Identifying Causes of Latency Inflation

The latency to a prefix can be inflated because of a problem either on the forward path from the CDN node to the prefix or on the reverse path back. To isolate the cause for inflation on either the forward path to or reverse path from an inflated prefix, *WhyHigh* uses data from BGP. Snapshots of the BGP routing table provide information on the AS path being used to route packets to all prefixes. A log of all the BGP updates tells us the other alternative paths available to each prefix. In addition, to obtain routing information at a finer granularity than an AS and to obtain visibility into the reverse path back from prefixes, *WhyHigh* also performs two kinds of active measurements—a traceroute [1] from the node to a destination in the prefix, and pings to intermediate routers seen on the traceroute. *WhyHigh* then uses these datasets to isolate whether the cause of inflation for an inflated prefix is on the forward or the reverse path.

To identify circuitousness along the forward path from a CDN node to a prefix, *WhyHigh* uses the sequence of locations traversed along the traceroute to the prefix. To identify circuitousness along the reverse path, *WhyHigh* uses three different techniques. First, it uses a significant RTT increase on a single hop of the traceroute to be an indicator of reverse path inflation. A packet that traverses any single link usually experiences a one-way latency of at most 40ms. We observed one-way link latencies to be less than 40ms on various long distance links that span thousands of miles (e.g., from coast to coast in the US, trans-Pacific, trans-Atlantic, and across Asia). Hence, an increase of 100ms or more in the RTT measured on successive hops of a traceroute likely indicates that the reverse path back from the latter hop is asymmetric. Second, the return TTL on the response from a probed interface provides an estimate of the length of the reverse path back from the interface. Therefore, a significant drop in the return TTL on pings to two successive interfaces on the traceroute implies that the reverse path back from at least one of the hops is asymmetric; comparison with forward TTLs tells us which one.

However, changes in RTT and return TTL across hops only provide hints about inflation on the reverse path. These do not suffice to identify the exact path taken by packets from the client to the node. To gain partial visibility into the reverse path, *WhyHigh* uses flow records gathered at border routers in Google's network; flow records are summaries of traffic forwarded by the router. Whenever records for flows to a client are available at a specific router, we can confirm that the reverse paths from that client pass through this router. We use this to compute the entry point of client traffic into the Google network. Therefore, if the path from a prefix to a node enters the Google network at a distant location from the node, we can infer circuitousness along the reverse path. In some cases, these techniques do not suffice for enabling *WhyHigh* to characterize the cause for inflation to a prefix to be either along the forward or reverse path. We, however, show later in Section 5 that such cases account for a small fraction of inflated prefixes.

4.3 Identifying Path Inflation Granularity

As seen previously in Figure 4, more than 20% of all prefixes are inflated, i.e., tens of thousands of prefixes are inflated. From the perspective of the CDN administrators, troubleshooting all of these prefixes individually is an intractable task. To make the administrator's task easier, *WhyHigh* leverages information from BGP paths to group prefixes potentially affected by a common underlying cause. For example, if a routing misconfiguration on a particular router is responsible for inflation on the path to a prefix, similar inflation is likely to be seen for other prefixes that have to traverse the same router.

Hence, *WhyHigh* merges prefixes into the largest set possible where all the prefixes in the set suffer from inflated latencies. The various granularities at which it attempts to group prefixes are: (i) prefixes sharing the same PoP-level path measured by traceroute,

---

[1] Traceroutes are performed with limited max TTL to ensure no probes reach the destination.
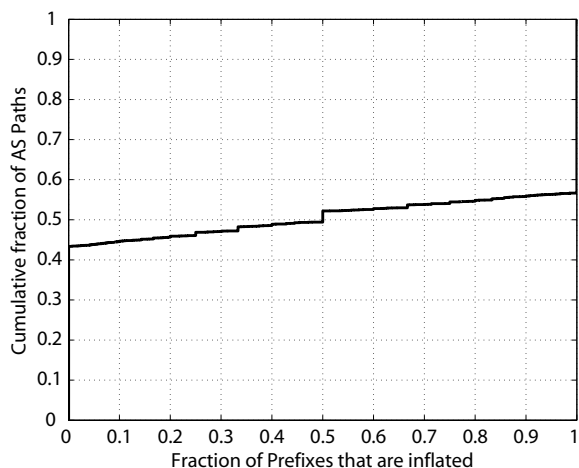
Figure 8: Fraction of prefixes served by an AS path that have inflated latencies. For most paths, either all prefixes have inflated latency or none of them do.
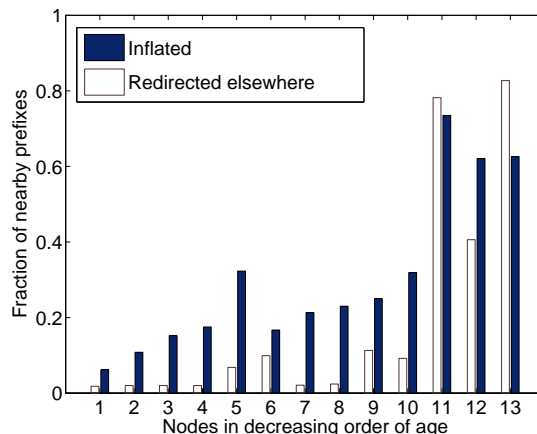

Figure 9: Of the prefixes geographically closest to a node, fraction that are inflated and fraction that are served elsewhere, i.e., not at the prefix's closest node.

(ii) prefixes sharing the same AS path and the same exit and entry PoPs out of and into Google's network, (iii) prefixes sharing the same AS path, and (iv) prefixes belonging to the same AS, in that order.

Figure 8 demonstrates the utility of grouping together prefixes by using data from one CDN node. For each AS path along which traffic is forwarded from this node, the figure plots the fraction of prefixes served by that AS path that are inflated. As seen in the figure, path inflation typically affects either all prefixes served by an AS path or none of the prefixes served by it. Hence, most instances of path inflation can be addressed at a higher granularity than prefixes, such as at the level of an AS path or an AS. This significantly reduces the problem space to be tackled by an administrator and lets her focus first on AS paths or ASes that account for a large number of inflated prefixes.

Grouping prefixes as above also helps *WhyHigh* isolate causes for inflation by providing information on what are possibly *not* the causes. For example, from a particular node, if the paths to all prefixes served by the AS path *path1* are inflated, and none of the paths to prefixes served by the AS path *path2* are, then one of the ASes on *path1* that is not on *path2* is likely to be responsible for the inflation.

4.4   Ranking Nodes

Another means by which *WhyHigh* helps administrators prioritize problems is by ranking the nodes in the CDN. At nodes afflicted by more widespread latency inflation, a single problem is typically the cause for inflation to several prefixes. Hence, following through with a single troubleshooting effort can have significant impact.

WhyHigh can rank order nodes in the CDN based on several metrics. A couple of example metrics quantifying a node's performance are—1) the fraction of nearby prefixes (which ought to be served at the node) that have inflated latencies, and 2) the fraction of nearby prefixes that are served elsewhere. A high value for either metric indicates a poorly performing node. Figure 9 plots both metrics for a subset of nodes in Google's CDN. The nodes are ordered in decreasing order of their age from left to right. This graph shows that new nodes are much more likely to have more widespread latency inflation issues, and unlike previously seen in Figure 3, a large fraction of prefixes near new nodes get redirected elsewhere. Therefore, the resources of an administrator are bet-

ter spent troubleshooting problems diagnosed by *WhyHigh* at these nodes.

4.5   Identifying Root Causes of Inflation

After having gathered all relevant information for every inflated prefix, *WhyHigh* attempts to pinpoint the set of causes for these problems. An administrator then goes over this list of causes, prioritizing them based on their impact, to verify them and to follow through on the proposed corrective actions.

The plausible causes that *WhyHigh* attributes to instances of latency inflation can largely be binned into four classes.

- **Lack of peering** [2]: When all available AS paths from a node to an inflated prefix are long even though the prefix's AS has presence in the same region as the node, having Google peer with the prefix's AS could fix the problem.
- **Limited bandwidth capacity**: In cases where the inflated prefix is in an AS that peers with Google and yet a circuitous path is being used to forward traffic from the node to the prefix, limited bandwidth capacity on the peering link between Google and the prefix's AS is the likely cause. To fix the latency inflation, more bandwidth needs to be provisioned to carry traffic from the node to all prefixes in the AS.
- **Routing misconfiguration**: *WhyHigh* attributes an instance of latency inflation to an error in routing configuration when the inflated prefix is in an AS which peers with Google, and though the direct path to the peering AS is being used, the reverse path back from the prefix is circuitous. The peering AS then needs to be contacted to correct its routing configuration.
- **Traffic engineering**: When an inflated prefix is not in an AS with whom Google peers, if a long AS path is being used from the node to the prefix even though alternate shorter paths exist, the latency inflation is likely due to traffic engineering. The network administrators who put in place the traffic engineering policies may not be aware of this side-effect, and hence, alerting them about the resulting latency inflation may help find a fix.

4.6   System Architecture

Figure 10 summarizes the steps involved in the *WhyHigh* pipeline. *WhyHigh* first gathers relevant logs from different data sources. It then associates each routable prefix with a host of diagnostic infor-

---

[2] We use the term *peering* loosely to refer to any kind of connection between two ASes.
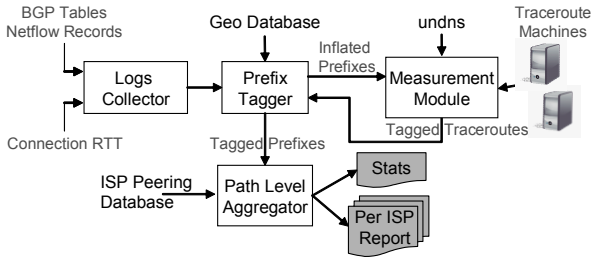
Figure 10: System architecture of *WhyHigh*.

AS Info	
AS Name, Country	Peering Locations
Statistics:	(a) Number of inflated prefixes
	(b) Number of affected connections
	(c) Expected improvement
Un-affected Paths	
Path:	(AS path1, Entry PoP1, Exit PoP1)
Prefixes:	P1, P2, . . .
Affected Paths	
Path:	(AS path2, Entry PoP2, Exit PoP2)
Prefixes:	P3, P4, . . .
Diagnostic Info	(a) Forward path inflation cases
	(b) Reverse path inflation cases
	(c) Culprit ASes
	(d) Sample traceroutes

Figure 11: Format of report produced at every Google node by *Why-High* for each AS to which inflated latency paths are observed.

mation based on data from these logs. This information includes 1) the RTT distribution observed by clients within a prefix, 2) the AS path to this prefix observed from all CDN nodes, 3) the geographic location of the prefix, and 4) the set of border routers within Google's network that have observed traffic to or from this prefix.

Once all prefixes have been tagged with relevant information, the *WhyHigh* system identifies prefixes having abnormally high latencies and performs active measurements from different nodes to these prefixes. This ensures that we only actively probe potential problem cases. The system then tags prefixes as having circuitous forward or reverse paths based on techniques described above.

WhyHigh outputs two kinds of information. First, it generates general statistics which help administrators monitor the performance of different nodes in the CDN. Second, it generates a per AS report as shown in Figure 11, which identifies inflated latency AS paths to prefixes in the AS and identifies the potential causes of inflation. In addition, to help in troubleshooting, *WhyHigh* also identifies good paths to the AS for comparison.

5. RESULTS

In this section, we present some results from our experience with running the *WhyHigh* system over the period of several months. First, we present a few representative instances of inflated latencies that were diagnosed using *WhyHigh*. We then present some results summarizing *WhyHigh*'s classification of detected problems and the improvement in client latencies as a result of following through on *WhyHigh*'s diagnosis.

5.1 Illustrative Example Cases

We employed *WhyHigh* to diagnose the causes for several instances of path inflation. Here, we present a few illustrative examples that are representative of the different types of causes we

```
ANC–Sprint–PhilISP1
ANC–Tiscalli–PhilISP1
ANC–Level3–NTT–PhilISP1
ANC–Level3–Teleglobe–PhilISP1
ANC–Level3–MCI Verizon–PhilISP1
Flag–PhilISP1
BTN–PhilISP1
```

Figure 12: AS paths used in troubleshooting Case 1. All paths go through the US en route to the destination in Philippines.

1.	1.1.1.1	0.116 ms
2.	1.1.1.2	0.381 ms
3.	japan1.**nrt**.google.com	145.280 ms
	(nrt → Narita, Japan)	
4.	exchangepoint.jp	146.327 ms
5.	router.**lax**.apacisp.com	262.749 ms
	(lax → Los Angeles, USA)	
6.	address1	509.779 ms

(a)

Prefix Length	Peering Point	AS path
/16	India	IndISP2–IndISP1
/18	SE Asia	ApacISP–Reliance–IndISP1
/18	Japan	ApacISP–Reliance–IndISP1

(b)

Figure 13: Data used in troubleshooting Case 2: (a) Extract of traceroute, and (b) AS paths received by Google.

encountered for path inflation.

Case 1: No peering, and all AS paths are long

WhyHigh reported that a node in southeast Asia consistently observed RTTs of more than 300ms to prefixes in PhilISP1 [3], an ISP in the Philippines. This was 200ms greater than the minimum RTT measured for that region. Given the distance between this node and the Philippines, such high RTTs are unexpected, even though Google does not directly connect with PhilISP1.

We examined the AS paths received at Google for PhilISP1's prefixes (shown in Figure 12). PhilISP1 splits its address space into several more specific prefixes and distributes these across its providers; we believe PhilISP1 is doing so to load-balance the traffic it receives across all of its peerings. PhilISP1's neighboring ASes on these AS paths have links with Google's providers only in the US. So, all the AS paths received by Google for reaching PhilISP1 traverse the US. The resultant circuitous routes on both the forward and reverse paths lead to RTTs greater than 300ms from the node in SE Asia to destinations in PhilISP1. Based on this information, *WhyHigh* diagnosed the cause for this case as the lack of peering between Google and PhilISP1.

Case 2: No peering, and shorter path on less specific prefix

A node in India measured RTTs above 400ms to destinations in IndISP1, an ISP in India. Though Google and IndISP1 do not directly peer, such high RTTs within a country are unreasonable.

WhyHigh revealed the following. As shown in Figure 13(a) [4], the traceroute from the Indian node to a destination in IndISP1 shows the forward path going through ApacISP, an ISP in the Asia-Pacific region, via Japan. ApacISP further worsens the inflation by forwarding the traffic through Los Angeles. This circuitous route

---

[3]Some of the AS names have been anonymized.

[4]Hostnames have been anonymized in all traceroutes.

	#Prefixes	#AS Paths	#ASes
Total	97852	23871	18754
Inflated	11862	1891	1510
Circuitous Fwd. Path	3865	1215	1038
Circuitous Rev. Path	4784	1579	1248
Misconfiguration	876	76	65
Limited Bandwidth	439	82	47
Lack of Peering	3776	710	456
Traffic Engineering	7410	1121	644

Table 1: *WhyHigh*'s classification of inflated paths.

```
1.  1.1.1.3                    0.339 ms
2.  1.1.1.4                    0.523 ms
3.  1.1.1.5                    0.670 ms
4.  japan2.nrt.google.com      0.888 ms
5.  exchangepoint.jp           1.538 ms
6.  router.japanisp.jp         117.391 ms
```
(a)

```
PING exchangepoint.jp
64 bytes from address2: ttl=252 time=1.814 msec
Estimated reverse path length = 4 hops

PING router.japanisp.jp
64 bytes from address3: ttl=248 time=102.234 msec
Estimated reverse path length = 8 hops
```
(b)

Figure 14: Data used in troubleshooting Case 4: (a) Extract of traceroute, and (b) Pings to routers at peering link.

causes path inflation for this case. *WhyHigh* reported better alternate paths for IndISP1's prefixes. In fact, as seen in Figure 13(b), we found that other than the path through ApacISP that was being used to carry the traffic, an AS path via IndISP2 was also received by Google in India. Since IndISP2 is an Indian ISP too, it surely connects with IndISP1 at some location in India. So, if this alternate AS path had been chosen instead to forward the traffic, we would observe much smaller latencies to end-hosts in IndISP1. However, this path is not an option since the prefixes associated with the AS paths through ApacISP are more specific than that associated with the AS path via IndISP2. On further investigation, this turned out to be due to the lack of capacity on the peering between IndISP2 and IndISP1.

Case 3: Peering, but longer paths on more specific prefixes
WhyHigh reported that connections served from the node in southeast Asia also experienced RTTs over 300ms to prefixes in PhilISP2, another ISP in the Phillipines. Unlike the previous two cases, in this instance RTTs are inflated even though Google connects with PhilISP2 in the region.

We analyzed the diagnosis information provided by *WhyHigh* for this case. Google receives a direct AS path to PhilISP2 for the prefixes that encompass all of PhilISP2's globally advertised address space. However, Google does not use this AS path for any traffic since PhilISP2 splits its address space into several smaller prefixes and announces these more specific prefixes to ASes other than Google. All of the AS paths through these other neighbors of PhilISP2 go through some AS in the US, such as AT&T or Level3. A router determines the route on which to forward traffic by performing a longest prefix match of the destination against the list of prefixes for which it received routes. Hence, the direct path from Google to PhilISP2 is never used. Routers forward all traffic to PhilISP2's prefixes over these AS paths that go through the US, resulting in extremely high end-to-end latencies.

Case 4: Peering, but inflated reverse path

In our final representative example, we look at a case reported by *WhyHigh* for JapanISP, an ISP in Japan that directly connects with Google. The RTT from the node in Japan to a set of prefixes in Japan hosted by JapanISPwas over 100ms. Again, this problem is worse than it first appears since Google connects with JapanISP in Japan.

To diagnose this problem, we looked at the traceroute, shown in Figure 14(a), from the node in Japan to an address in one of the

problematic prefixes. The RTT increased from around 1ms to over 100ms across the hops at either end of the peering link between Google and JapanISP, indicating inflation on the reverse path back from JapanISP. An increase of 4 hops in the reverse path length estimated by pings to either end of the peering link further substantiates the reverse path asymmetry (see Figure 14(b)). Since Google also connects with JapanISP on the US west coast, we suspected that JapanISP was handing packets back to Google there. This was confirmed by analysis of router flow records. We believe this situation is a legacy of the fact that Google's presence was originally limited to California, and that many Asian ISPs consider Google to be only present in the US and prefer to send traffic to all Google nodes via their PoPs in the US.

5.2 Resolution of Cases

We presented the diagnostic information from *WhyHigh* for these and other similar instances of path inflation to the network operations staff at Google.

To solve the reverse path inflation back from JapanISP, Google's network admins restricted the routing announcement for the Google node in question to be available only in Japan by advertising more specific prefixes at the the PoP in Japan where Google peers with JapanISP. This change reduced the RTT from over 100 milliseconds to a few milliseconds on the paths to JapanISP we were troubleshooting. Also, *WhyHigh* reported that the entry and exit points for paths from JapanISP were both now in Japan. Another option to address this case would have been to influence JapanISP to perform early-exit routing at their PoP in Japan, instead of having a higher preference to route traffic to Google via the US.

The network admins informed us that PhilISP2 advertises a less specific prefix to Google than to its other neighbors because of the limited capacity on the peering link between Google and PhilISP2. Once this link was upgraded, *WhyHigh* no longer detected inflated paths to PhilISP2.

5.3 Summarizing use of *WhyHigh*

Table 1 presents a breakdown of the various causes into which *WhyHigh* classified the latency inflation problems observed in our RTT dataset. The number of cases to be investigated reduced by almost an order of magnitude when considering the set of affected AS paths and ASes, rather than prefixes. While most of the cases were diagnosed as being due to lack of peering or traffic engineering, there were also a significant number of cases affected by routing misconfigurations and limited bandwidth on peering links.

We analyze one of the causes—lack of peering—in more detail. Given a node with inflated paths, we seek to understand what fraction of these instances of inflation are in ISPs reached via transit ASes versus those in ISPs with whom Google peers. This is important because peering relations with ISPs are expensive to establish and should be utilized efficiently. Also, if an ISP with significant
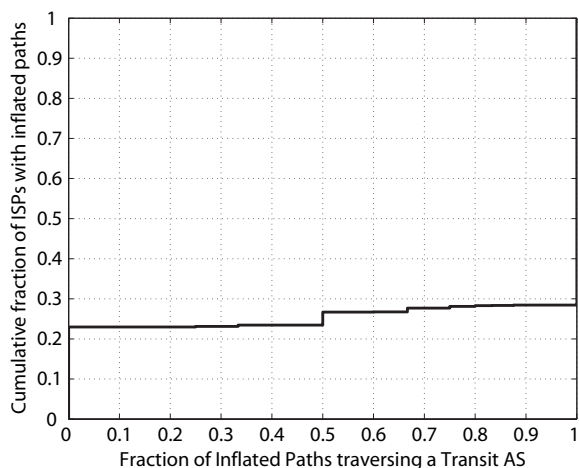
Figure 15: Fraction of inflated AS paths to an AS that goes via a transit AS. At this particular node, around 70% of ISPs experiencing path inflation have all paths going over transit, indicating that peering could be a potential solution.
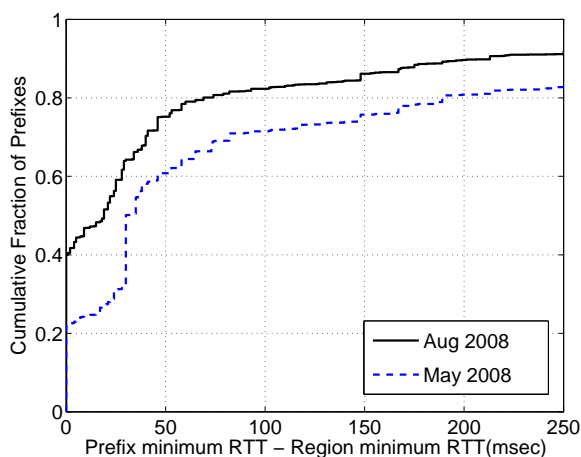

Figure 16: Improvement in latencies for prefixes served by a node in South America from May 2008 to August 2008.

traffic has inflated latencies because of lack of peering, i.e., the path traverses multiple transit providers, it maybe worthwhile to establish new peering relations.

Figure 15 plots the fraction of inflated latency AS paths to an ISP that traverse a transit provider. A path is considered to have inflated latency if all the prefixes served by that path experience latency inflation. We see that for this particular node, around 70% of ISPs experiencing path inflation have all their inflated AS Paths via transit providers. Lack of peering relations is the likely reason for inflation on paths to ISPs in this set. We also see that around 22% of the ISPs with inflated paths are so even though all paths utilize the direct peering with Google. These cases are important to debug because significant time and effort is involved in establishing peering relations. Also, it is easier for administrators to contact peer ISPs to fix misconfigurations in their network.

Similar to the example cases outlined above, several other instances of inflation have been fixed, and the resolution of many instances we diagnosed is in progress. Since several thousand prefixes are affected by inflation, we have been prioritizing cases in the order in which Google receives traffic corresponding to them. To provide an example of the utility of *WhyHigh*, Figure 16 presents

1.	1.1.1.6	0.186 ms
2.	1.1.1.7	0.428 ms
3.	node.seasia.google.com	1.727 ms
4.	router1.nrt.asianisp.net	35.600 ms
5.	router2.nrt.asianisp.net	35.644 ms
6.	router.tpe.taiwanisp.net	513.077 ms

Figure 17: Extract of traceroute from an inconclusive case of inflation observed from a Google node in Southeast Asia to a destination in Taiwan.

the improvement in path latencies for prefixes served by a Google node in South America—a node where we have focused a lot of our efforts at troubleshooting since the extent of the problems here were significantly worse compared to elsewhere. In May 2008, as many as 40% of prefixes that were being redirected to this node were experiencing latencies more than 50ms in excess of other prefixes in the region. The information provided by *WhyHigh* has helped Google's network operations staff to work with ISPs in the region and significantly improve path latencies. More recent data from August 2008 shows that the fraction of prefixes being served by this South American node that suffer over 50ms of inflation is now reduced to 22%.

6. LIMITATIONS

After having looked at examples in Section 5 of cases that *Why-High* was successfully able to diagnose, we now consider an example where *WhyHigh* was inconclusive. We observed RTTs in excess of 500ms from a node in Southeast Asia to a bunch of prefixes hosted by an ISP in Taiwan, which does not peer with Google. The traceroute to an address in one of the problematic prefixes (shown in Figure 17) revealed that the path traversed another ISP in Asia enroute to the destination ISP in Taiwan. Note that RTT from the 5th hop in Narita, Japan to the 6th hop in Taipei jumps from 35ms to 513ms. The distance from Narita to Taipei does not account for such a steep rise in RTTs. Therefore, we expected inflation to be on the reverse path back from the destination ISP in Taiwan. However, we observed no significant jump in return TTL values in pings to successive hops on the traceroute, and we found no flow records to confirm a circuitous reverse path. As a result, we lacked sufficient evidence to confirm our suspicion of inflation along the reverse path.

We present this example to illustrate that though *WhyHigh* helps in diagnosing the cause of several instances of path inflation, it is not perfect. This is a consequence of *WhyHigh*'s partial view of Internet routing. A predominant reason is the significant presence of asymmetric paths in the Internet [15, 8]. Though we were able to partially tackle route asymmetry by using flow records from routers, in order to more precisely troubleshoot problems, *WhyHigh* needs the ability to gather information about the reverse path back from clients to Google's nodes. An option is to own a measurement node within the client network which, if possible, is an expensive alternative.

WhyHigh's incomplete view of the network also stems from the fact that the finest granularity of information it uses is from traceroute. Traceroutes yield path information only at the IP routing layer, i.e., layer 3 of the networking stack. However, path inflation could occur below layer 3, e.g., in MPLS tunnels, and hence may not be explainable by the geographic locations of traceroute hops, as seen in the example presented in Figure 17.

7. RELATED WORK

Content Distribution Networks (CDNs) have emerged as a com-

monly used platform to serve static content from the Web. Popular examples include Akamai [1] and Coral [6]. In these systems, different clients are redirected to different servers in an attempt to provide load balancing and to improve client performance. The common approach [7] employed to improve performance is to redirect a client to the server to which it has the least latency. The underlying assumption is that in the presence of a geographically distributed set of servers, latency-based redirection will direct the client to a nearby server to which the client has a low RTT. Our large scale measurement dataset from Google's CDN nodes to clients all across the Internet shows that this is not necessarily true for all clients. Routing inefficiencies and queueing of packets both lead to significant inflation of path latencies.

Recently the performance of two large-scale CDNs—Akamai and Limelight—has been evaluated in [9]. The authors are able to chart different nodes in these networks from outside the CDNs. They also compare the delay performance of the two CDNs and establish that Akamai yields better median delay. In addition, based on the computed location of Akamai servers, they suggest alternative locations for Limelight servers to improve performance. On the other hand, WISE [19] aims to answer how changes to the CDN's deployment would impact the performance it offers to clients. Our work is complementary to both. Rather than studying how to augment a CDN's deployment or understanding the impact of a change in node configuration, we detect the problems that hinder an existing CDN deployment from offering the best performance possible. Before undertaking the significant effort and investment required to setup new nodes, *WhyHigh* enables the administrator to characterize the performance of an existing deployment and to diagnose routing inefficiencies that hinder the CDN from delivering the best performance possible to clients.

Path inflation in the Internet has been studied [17] previously, and its causes have been traced predominantly to inter-domain routing policies. Extensions to BGP have been proposed [13] to provide ISPs the ability to cooperatively choose globally optimal routes without revealing their local routing policies. We find path inflation to be much more pronounced than observed previously. Even when considering RTTs from prefixes to nodes that are predominantly nearby, we find that more than 20% of prefixes experience inflation greater than 50ms. In contrast, less than 5% of paths were estimated to have more than 25ms inflation in prior work [17].

Overlay routing systems [16, 2] have attempted to circumvent inflated paths by routing through end-hosts that serve as relays. Instead, we focus on fixing instances of path inflation by identifying the underlying cause. Though the predominant causes of path inflation are known, we are the first to study how the cause for inflation on any given path can be identified. *WhyHigh* diagnoses the cause for each instance of inflated latency, thus providing network administrators with the steps they need to take to improve the performance of their clients.

WhyHigh uses active measurements, such as traceroutes and pings, if necessary. Traceroutes are a common network diagnostic tool, and have been used primarily to diagnose reachability issues and to pinpoint such problems to a particular AS, router, or link, e.g., PlanetSeer [21], Hubble [11], and [22]. We use traceroutes instead for performance diagnosis of paths, similar to iPlane [12] and Netdiff [14]. *WhyHigh* uses the geographic location of routers observed in the output of traceroute to determine if the path is circuitous. To do so, we use router DNS name to location mappings developed as part of the Rocketfuel project [18]. *WhyHigh* also attempts to group together prefixes affected by the same underlying cause, an approach used previously to diagnose BGP routing instabilities [5, 20].

8. CONCLUSIONS AND FUTURE WORK

Replicating content across a geographically distributed set of servers and redirecting every client to the server with least latency is commonly expected to significantly improve client performance. However, using data from Google's CDN infrastructure that serves over 170K prefixes across the Internet, we find that routing inefficiencies and packet queueing can greatly undermine the potential improvements in RTT that a CDN can yield. To aid administrators in the task of debugging instances of high RTTs, we have built the *WhyHigh* system which first identifies affected prefixes, then prioritizes the several tens of thousands of inflated prefixes, and finally diagnoses the causes underlying these cases. We see significant improvements in client latencies offered by Google's CDN from the use of *WhyHigh* over the course of a year.

Given the large fraction of prefixes affected by path inflation issues, we are working on better visualization of our RTT data; clustering of problems with the same underlying cause is vital in our effort to debug Internet routing. Many questions still remain open on the cause for poor latencies we observe from clients to nearby nodes, e.g., "where are packets being queued to result in the congestion overhead we observe?", and we look to answer these in the future to further improve client performance.

9. REFERENCES

[1] Akamai, Inc. home page. http://www.akamai.com.

[2] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, 2001.

[3] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *INFOCOM*, 2000.

[4] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *IMC*, 2007.

[5] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs. Locating Internet routing instabilities. In *SIGCOMM*, 2004.

[6] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *NSDI*, 2004.

[7] M. J. Freedman, K. Lakshminarayanan, and D. Mazieres. OASIS: Anycast for any service. In *NSDI*, 2006.

[8] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker. On routing asymmetry in the Internet. In *Autonomic Networks Symposium in Globecom*, 2005.

[9] C. Huang, A. Wang, J. Li, and K. W. Ross. Measuring and evaluating large-scale CDNs. In *IMC*, 2008.

[10] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards IP geolocation using delay and topology measurements. In *IMC*, 2006.

[11] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson. Studying black holes in the Internet with Hubble. In *NSDI*, 2008.

[12] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, 2006.

[13] R. Mahajan. *Practical and Efficient Internet Routing with Competing Interests*. PhD thesis, University of Washington, 2005.

[14] R. Mahajan, M. Zhang, L. Poole, and V. Pai. Uncovering performance differences among backbone ISPs with Netdiff. In *NSDI*, 2008.

[15] V. Paxson. End-to-end routing behavior in the Internet. *ToN*, 1997.

[16] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: a case for informed Internet routing and transport. *IEEE Micro*, 1999.

[17] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *SIGCOMM*, 2003.

[18] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *ToN*, 2004.

[19] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering what-if deployment and configuration questions with WISE. In *SIGCOMM*, 2008.

[20] J. Wu, Z. M. Mao, , and J. R. J. Wang. Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network. In *NSDI*, 2005.

[21] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *OSDI*, 2004.

[22] Y. Zhang, Z. M. Mao, and M. Zhang. Effective diagnosis of routing disruptions from end systems. In *NSDI*, 2008.