LatLong: Diagnosing Wide-Area Latency Changes for CDNs

Yaping Zhu, Benjamin Helsley, Jennifer Rexford, Aspi Siganporia, and Sridhar Srinivasan

***Abstract*—Minimizing user-perceived latency is crucial for Content Distribution Networks (CDNs) hosting interactive services. Latency may increase for many reasons, such as interdomain routing changes and the CDN's own load-balancing policies. CDNs need greater visibility into the causes of latency increases, so they can adapt by directing traffic to different servers or paths. In this paper, we propose a tool for CDNs to diagnose large latency increases, based on *passive* measurements of performance, traffic, and routing. Separating the many causes from the effects is challenging. We propose a *decision tree* for classifying latency changes, and determine how to distinguish traffic shifts from increases in latency for existing servers, routers, and paths. Another challenge is that network operators group related clients to reduce measurement and control overhead, but the clients in a region may use multiple servers and paths during a measurement interval. We propose metrics that quantify the latency contributions across *sets* of servers and routers. Based on the design, we implement the LatLong tool for diagnosing large latency increases for CDN. We use LatLong to analyze a month of data from Google's CDN, and find that nearly 1% of the daily latency changes increase delay by more than 100 msec. Note that the latency increase of 100 msec is significant, since these are daily averages over groups of clients, and we only focus on latency-sensitive traffic for our study. More than 40% of these increases coincide with interdomain routing changes, and more than one-third involve a shift in traffic to different servers. This is the first work to diagnose latency problems in a large, operational CDN from purely passive measurements. Through case studies of individual events, we identify research challenges for managing wide-area latency for CDNs.**

***Index Terms*—Network diagnosis, latency increases, content distribution networks (CDNs).**

I. Introduction

CONTENT Distribution Networks (CDNs) offer users access to a wide variety of services, running on geographically distributed servers. Many web services are delay-sensitive interactive applications (e.g., search, games, and collaborative editing). CDN administrators go to great lengths to minimize user-perceived latency, by overprovisioning server resources, directing clients to nearby servers, and shifting traffic away from overloaded servers. Yet, CDNs are quite vulnerable to increases in the *wide-area* latency between their servers and the clients, due to interdomain routing changes or congestion in other domains. The CDN administrators need to detect and diagnose these large increases in round-trip time, and adapt to alleviate the problem (e.g., by directing clients to a different front-end server or adjusting routing policies to select a different path).

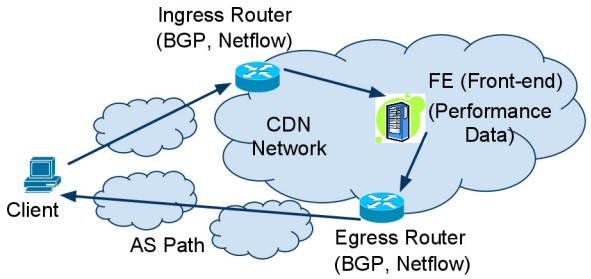


Fig. 1. CDN architecture and measurements.

To detect and diagnose latency problems, CDNs could deploy a large-scale active-monitoring infrastructure to collect performance measurements from synthetic clients all over the world. Instead, this paper explores how CDNs can diagnose latency problems based on measurements they can readily and efficiently collect—*passive* measurements of performance [1], traffic [2], and routing from their own networks. Our goal is to design the system to maximize the information the CDN can glean from these sources of data. By joining data collected from different locations, the CDN can determine where a client request enters the CDN's network, which front-end server handles the request, and what egress router and interdomain path carry the response traffic, as shown in Figure 1. Using this data, we analyze changes in wide-area latency between the clients and the front-end servers; the rest of the user-perceived latency, between the front and back-end servers, is already under the CDN's direct control.

Finding the *root cause* of latency increases is difficult. Many factors can contribute to higher delays, including internal factors like how the CDN selects servers for the clients, and external factors such as interdomain routing changes. Moreover, separating cause from effect is a major challenge. For example, directing a client to a different front-end server naturally changes where traffic enters and leaves the network, but the routing system is not to blame for any resulting increase in latency. After detecting large increases in latency, our classification must first determine whether client requests shifted to different front-end servers, or the latency to reach the existing servers increased. Only then can we analyze *why* these changes happened. For example, the front-end server may change because the CDN determined that the client is closer to

Manuscript received August 26, 2011; revised November 30, 2011 and March 20, 2012; accepted May 3, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was C. Wang.

Y. Zhu and J. Rexford are with Princeton University (e-mail: {yapingz, jrex}@cs.princeton.edu).

B. Helsley, A. Siganporia, and S. Srinivasan are with Google Inc. (e-mail: {bhelsley, aspi, sridhars}@google.com).

Digital Object Identifier 10.1109/TNSM.2012.12.110180

a different server, or because a load-balancing policy needed to shift clients away from an overloaded server. Similarly, if the round-trip time to a specific server increases, routing changes along the forward or reverse path (or both!) could be responsible.

The *scale* of large CDNs also introduces challenges. To measure and control communication with hundreds of millions of users, CDNs typically group clients by prefix or geographic region. For example, a CDN may collect round-trip times and traffic volumes by IP prefix, or direct clients to front-end servers by region. During any measurement interval, a group of clients may send requests to *multiple* front-end servers, and the traffic may traverse *multiple* ingress and egress routers. Thus, in order to analyze the latency increases for groups of requests, we need to define the metrics to distinguish the changes from an individual router or server.

In designing our tool LatLong for classifying large latency increases, we make the following contributions:

Decision tree for separating cause from effect: A key contribution of this paper is that we determine the causal relationship among the various factors which lead to latency increases. We propose a *decision tree* for separating the causes of latency changes from their effects, and identify the data sets needed for each step in the analysis. We analyze the measurement data to identify suitable thresholds to identify large latency changes and to distinguish one possible cause from another.

Metrics to analyze over *sets* of servers and routers: Our tool LatLong can analyze latency increases and traffic shifts over *sets* of servers and routers. For all potential causes of the latency increase in the decision tree, we propose metrics to quantify the contribution of the latency increases. For each potential cause, we define the metric to quantify the contribution of latency increases by a single router or server, as well as a way to summarize the contributions across all routers and servers.

Deployment of LatLong in Google's CDN: We apply our tool to one month of traffic, performance, and routing data from Google's CDN, and focus our studies on the large latency increases which last long and affect a large number of clients. Note that our tool could be applied to study the latency increases at any granularity. We focus on the large increases, because these are the events which causes serious performance degradation for the clients. We determine 100 msec as the threshold of large latency increase, because it is significant given that this number is the daily average aggregated from group of clients. We also focus on the latency-sensitive traffic for interactive applications for our study, which does *not* include video traffic (e.g., YouTube). We identified that nearly 1% of the daily latency changes increase delay by more than 100 msec. Our results show that 73.9% of these large increases in latency were explained (at least in part) by a large increase in latency to reach an existing front-end server, with 42.2% coincided with a change in the ingress router or egress router (or both!); around 34.7% of the large increases of latency involved a significant shift of client traffic to different front-end servers, often due to load-balancing decisions or changes in the CDN's own view of the closest server.

Case studies to highlight challenges in CDN management: We present several events in greater detail to highlight the challenges of measuring and managing wide-area performance. These case studies illustrate the difficulty of building an accurate latency-map to direct clients to nearby servers, the extra latency client experience when flash crowds force some requests to distant front-end servers, and the risks of relying on AS path length as an indicator of performance. Although many of these problems are known already, our case studies highlight that these issues arise in practice and are responsible for very large increases in latency affecting real users.

Our tool is complementary to the recent work on WhyHigh [3]. WhyHigh uses *active* measurements, combined with routing and traffic data, to study *persistent* performance problems where some clients in a geographic region have much higher latency than others. In contrast, we use *passive* measurements to analyze large latency *changes* affecting entire groups of clients. The *dynamics* of latency increases caused by changes in server selection and inter-domain routing are not studied in the work of WhyHigh.

The rest of the paper is organized as follows. Section II provides an overview of the architecture of the Google CDN, and the datasets we gathered. Section III describes our design of LatLong using decision-tree based classification. Section IV presents a high-level characterization of the latency changes in the Google's CDN, and identifies the large latency events we study. Next, we present the results of classification using LatLong in Section V, followed by several case studies in Section VI. Then, we discuss the future research directions in Section VII, and present related work in Section VIII. Finally, we conclude the paper in Section IX.

II. Google's CDN and Measurement Data

In this section, we first provide a high-level overview of the network architecture of Google's CDN. Then, we describe the measurement dataset we gathered as the input of our tool.

A. Google's CDN Architecture

The infrastructure of Google's CDN consists of many servers in the data centers spread across the globe. The client requests are first served at a front-end (FE) server, which provides caching, content assembly, pipelining, request redirection, and proxy functions for the client requests. To have greater control over network performance, CDN administrators typically place front-end servers in managed hosting locations, or ISP points of presence, in geographic regions nearby the clients. The client requests are terminated at the FEs, and (when necessary) served at the backend servers which implement the corresponding application logic. Inside the CDN's internal network, servers are connected by the routers, and IP packets enter and leave the network at edge routers that connect to neighboring ISPs.

Figure 1 presents a simplified view of the path of a client request. A client request is directed to an FE, based on proximity and server capacity. Each IP packet enters the CDN network at an *ingress router* and travels to the chosen FE. After receiving responses from the back-end servers, the FE directs response traffic to the client. These packets leave the CDN at an *egress router* and follow an *AS path* through one

TABLE I
MEASUREMENTS OF WIDE-AREA PERFORMANCE, TRAFFIC, AND ROUTING

Data Set	Collection Point	Logged Information
Performance	front ends (FEs)	(client /24 prefix, country, RPD, average RTT)
Traffic	ingress routers	(client IP address, FE IP address, bytes-in)
	egress routers	(FE IP address, client IP address, bytes-out)
Routing	egress routers	(client IP prefix, AS path)
Joint data	(client IP prefix, FE, RPD, RTT, {ingress, bytes-in}, {egress, AS path, bytes-out})	

or more Autonomous Systems (ASes) en route to the client. The user-perceived latency is affected by several factors: the location of the servers, the path from the client to the ingress router, and the path from the egress router back to the client. From the CDN's perspective, the visible factors are: the ingress router, the selection of the servers, the egress router, and the AS path.

Like many CDNs, Google uses DNS to direct clients to front-end servers, based first on a *latency map* (preferring the FE with the smallest network latency) and second on a *load-balancing* policy (that selects another nearby FE if the closest FE is overloaded) [4]. To periodically construct the latency map, the CDN collects round-trip statistics by passively monitoring TCP transfers to a subset of the IP prefixes. In responding to a DNS request, the CDN identifies the IP prefix associated with the DNS resolver and returns the IP address of the selected FE, under the assumption that end users are relatively close to their local DNS servers. Changes in the latency map can lead to shifts in traffic to different FEs. The latency between the front-end and back-end servers is a known and predictable quantity, and so our study focuses on the network latency—specifically, the round-trip time—between the FEs and the clients.

B. Passive Measurements of the CDN

The measurement data sets, which are routinely collected at the servers and routers, are summarized in Table II. The three main datasets—performance, traffic, and routing measurements—are collected by different systems. The measurement data gathered is composed of latency sensitive traffic for the interactive applications. We do *not* include the video traffic(e.g., YouTube) for our study, because that is latency insensitive.

Client performance (at the FEs): The FEs monitor the round-trip time (RTT) for a subset of the TCP connections by measuring the time between sending the SYN-ACK and receiving an ACK from the client. In cases when the SYN-ACK or ACK packet is lost, this SYN-ACK RTT value would be invalid. In these cases, the RTT for data transfers in the same TCP connection would be used instead. These measurements capture the propagation and queuing delays along both the forward and reverse paths to the clients. Each FE also counts the number of requests, producing a daily summary of the round-trip time (RTT) and the requests per day (RPD) for each /24 IP prefix. Each /24 prefix is associated with a specific country, using an IP-geo database. We use it to group prefixes in nearby geographical regions for our study.

Netflow traffic (at edge routers): The edge routers collect traffic measurements using Netflow [2]. The client is the source address for incoming traffic and the destination address for outgoing traffic; similarly, the FE is the destination for incoming traffic, and the source for outgoing traffic. Netflow performs packet sampling, so the traffic volumes are estimates after correcting for the sampling rate. This leads to records that summarize traffic in each fifteen-minute interval, indicating the client IP address, front-end server address, and traffic volume. Traffic for a single client address may be associated with multiple routers or FEs during the interval. The Netflow data we use chooses bytes as the metric, because it represents the traffic volume we care about. However, our techniques could also be applied to analyze the number of flows. We do not see any significant distinction between using the bytes versus the flows as the metric.

BGP routing (at egress routers): The edge routers also collect BGP routing updates that indicate the sequence of Autonomous Systems (ASes) along the path to each client IP prefix. (Because BGP routing is destination based, the routers cannot collect similar information about the forward path from clients to the FEs.) A dump of the BGP routing table every fifteen minutes, aligned with the measurement interval for the Netflow data, indicates the AS-PATH of the BGP route used to reach each IP prefix from each egress router.

Joint data set: The joint data set used in our analysis combines the performance, traffic, and routing data, using the client IP prefix and FE IP address as keys in the join process. First, the traffic and routing data at the egress routers are joined by matching the client IP address from the Netflow data with the longest-matching prefix in the routing data. Second, the combined traffic and routing data are aggregated into summaries and joined with the performance data, by matching the /24 prefix in the performance data with the longest-matching prefix from the routing data. The resulting joint data set captures the traffic, routing, and performance for each client IP prefix and front-end server, as summarized in Table II. The data set is aggregated to prefix level. In addition, the data do not contain any user-identifiable information (such as packet payloads, timings of individual requests, etc.) The data set we study is based on a sample, and does not cover all of the CDN network.

The data have some unavoidable limitations, imposed by the systems that collect the measurements: the performance data does not indicate which ingress and egress router were used to carry the traffic, since the front-end servers do not have access to this information. This explains why the joint data set has a *set* of ingress and egress routers. Fortunately, the Netflow measurements allow us to estimate the request rate for the individual ingress routers, egress routers, and AS paths from the observed traffic volumes; however, we cannot

directly observe how the RTT varies based on the choice of ingress and egress routers. Still, the joint data set provides a wealth of information that can shed light on the causes of large latency increases.

Latency map, and front-end server capacity and demand: In addition to the joint data set, we analyze changes to the latency map used to drive DNS-based server selection, as discussed in more detail in Section III-B. We also collect logs of server capacity and demand at all front-end servers. We use the logs to determine whether a specific FE was overloaded at a given time (when the demand exceeded capacity, and requests were load balanced to other front-end servers).

III. Design of the LatLong Tool

Analyzing wide-area latency increases is difficult, because multiple inter-related factors can lead to higher round-trip times. Also, our analysis should account for the fact that clients may direct traffic to *multiple* front ends, either because the front-end server changes or because different clients in the same region use different front-end servers.

In this section, we present the design of the decision tree which LatLong uses to analyze latency increases, as illustrated in Figure 2. We propose metrics for distinguishing FE changes from latency changes that affect individual FEs. Then, we describe the techniques to identify the cause of FE changes (the latency map, or load balancing). Lastly, we present the method to correlate the latency increases that affect individual FEs with routing changes. The classification of our tool is general, and the method is not dependent on specific way to aggregate users or specific timescale. Therefore, we can support to diagnose latency changes at different granularities (e.g., different ways to aggregate users, and different timescales).

Table III summarizes the notation used in this paper.

A. Front-End Server Change vs. Latency Increase

The average round-trip time could increase for one of two main reasons:

- **Front-end server changes (ΔFE):** The clients switch from one front-end server to another, where the new server used has a higher RTT. This change could be caused by an FE failure or a change in the CDN's server-selection policies, as shown in the upper branch of Figure 2.
- **Front-end latency changes (ΔLat):** The clients could continue using the same FE, but have a higher RTT for reaching that server. The increased latency could stem from changes along the forward or reverse path to the client, as shown in the lower branch of Figure 2.

The analysis is difficult because a group of clients could contact multiple front-end servers, and the RTT and RPD for each server changes. Correctly distinguishing all of these factors requires grappling with *sets* of front-ends and *weighting* the RTT measurements appropriately.

The average round-trip time experienced by the clients is the average over the requests sent to multiple front-ends, each with its own average round-trip time. For example, consider a region of clients experiencing an average round-trip time of RTT_1 at time 1, with a request rate of RPD_{1i} and round-trip time RTT_{1i} for each front-end server i. Then,

$$RTT_1 = \sum_i RTT_{1i} * \frac{RPD_{1i}}{RPD_1}$$

where $RPD_1 = \sum_i RPD_{1i}$ is the total number of requests from that region, across all front-end servers, for time period 1. A similar equation holds for the second time period, with the subscripts changed to consider round-trip times and request rates at time 2.

The increase in average round-trip time from time 1 to time 2 (i.e., $\Delta RTT = RTT_2 - RTT_1$) is, then,

$$\Delta RTT = \sum_i \left(RTT_{2i} * \frac{RPD_{2i}}{RPD_2} - RTT_{1i} * \frac{RPD_{1i}}{RPD_1} \right)$$

The equation shows how the latency increases could come either from a higher round-trip time for the same server (i.e., $RTT_{2i} > RTT_{1i}$) or a shift in the fraction of requests directed to each FE (i.e., RPD_{2i}/RPD_2 vs. RPD_{1i}/RPD_1), or both.

To tease these two factors apart, consider one FE i, and the term inside the summation. We can split the term into two parts that sum to the same expression, where the first captures the impact on the round-trip time from traffic shifting toward front-end server i:

$$\Delta FE_i = RTT_{2i} * \left(\frac{RPD_{2i}}{RPD_2} - \frac{RPD_{1i}}{RPD_1} \right)$$

where ΔFE_i is high if the fraction of traffic directed to front-end server i increases, or if the round-trip time is high at time 2. The second term captures the impact of the latency to front-end server i increasing:

$$\Delta Lat_i = (RTT_{2i} - RTT_{1i}) * \frac{RPD_{1i}}{RPD_1}$$

where the latency is weighted by the fraction of requests directed to front-end server i, to capture the relative impact of this FE on the total increase in latency. Through simple algebraic manipulation, we can show that

$$\Delta RTT = \sum_i (\Delta FE_i + \Delta Lat_i).$$

As such, we can quantify the contribution to the latency change that comes from shifts between FEs:

$$\Delta FE = \sum_i \Delta FE_i / \Delta RTT$$

and latency changes for individual front-end servers

$$\Delta Lat = \sum_i \Delta Lat_i / \Delta RTT$$

where the factors sum to 1. For example, if the FE change contributes 0.85 and the latency change contributes 0.15, we can conclude that the latency increase was primarily caused by a traffic shift between front-end servers. If the FE change contributes -0.1 and the latency change contributes 1.1, we can conclude that the latency increase was due to an increase in latency to reach the front-end servers rather than a traffic shift; if anything, the -0.1 suggests that some traffic shifted to FEs with *lower* latency, but this effect was dwarfed by one or more FEs experiencing an increase in latency.

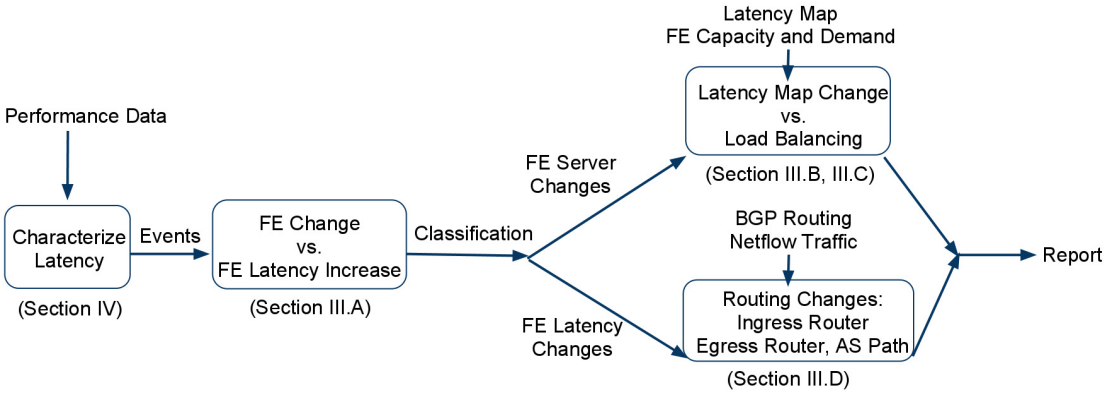


Fig. 2. LatLong system design: classification of large latency changes.

TABLE II
SUMMARY OF KEY NOTATION

Symbol	Meaning
RTT_1, RTT_2	round-trip time for a client region at time 1 and time 2
ΔRTT	change in RTT from time 1 to time 2 (i.e., $RTT_2 - RTT_1$)
RTT_{1i}, RTT_{2i}	round-trip time for requests to FE_i at time 1 and time 2
RPD_1, RPD_2	requests for a client region at time 1 and time 2
RPD_{1i}, RPD_{2i}	requests to FE_i at time 1 and time 2
ΔFE_i	latency change contribution from traffic shifts at FE_i
ΔLat_i	latency change contribution from latency changes at FE_i
ΔFE	latency change contribution from traffic shifts at all FEs
ΔLat	latency change contribution from latency changes at all FEs
r_{1i}, r_{2i}	fraction of requests served at FE_i predicted by the latency map at time 1 and time 2
$\Delta LatMap$	fraction of requests shifting FEs predicted by the latency map
$\Delta FEDist$	actual fraction of requests shifting FEs
$LoadBalance_1$	fraction of requests shifting FEs by the load balancer at time 1
$\Delta LoadBal$	difference of the fraction of requests shifting FEs by the load balancer from time 1 to time 2
$\Delta Ingress$	fraction of the traffic shifting ingress router at a specific FE
$\Delta EgressASPath$	fraction of the traffic shifting (egress router, AS path) at a specific FE

In the following subsections, we present the method to identify the causes of the FE changes: the latency map and load balancing.

B. Front-End Changes by the Latency Map

Google CDN periodically constructs a latency map to direct clients to the closest front-end server. The CDN constructs the latency map by measuring the round-trip time for each /24 prefix to different front-end servers, resulting in a list mapping each /24 prefix to a single, closest FE. From the latency map, we can compute the target distribution of requests over the front-end servers for groups of co-located clients in two time intervals. To combine this information across all /24 prefixes in the same region, we weight by the requests per day (RPD) for each /24 prefix. This results in a distribution of the fraction of requests r_{1i} from the client region directed to front-end server i, at time 1.

As the latency map and the request rates change, the region may have a different distribution $\{r_{2i}\}$ at time 2. To analyze changes in the latency map, we consider the fraction of requests that should shift to different front-end servers:

$$\Delta LatMap = \sum_i |r_{2i} - r_{1i}|/2$$

Note that we divide the difference by two, to avoid double counting the fraction of requests that move away from one FE (i.e., $r_{2i} - r_{1i}$ decreasing for one front-end server i) and towards another (i.e., $r_{2i} - r_{1i}$ increasing for some other front-end server).

C. Front-End Changes by Load Balancing

In practice, the actual distribution of requests to front-end servers does not necessarily follow the latency map. Some FEs may be overloaded, or unavailable due to maintenance. To understand how the traffic distribution changes in practice, we quantify the changes in front-end servers as follows:

$$\Delta FEDist = \sum_i \left| \frac{RPD_{2i}}{RPD_2} - \frac{RPD_{1i}}{RPD_1} \right| /2$$

That is, we calculate the fraction of requests to FE i at time 1 and time 2, and compute the difference, summing over all front-end servers. As with the equation for $\Delta LatMap$, we divide the sum by two to avoid double counting shifts away from one front-end server and shifts toward another.

The differences are caused by the CDN's own load-balancing policy, which directs traffic away from busy front-end servers. This may be necessary during planned maintenance. For example, an FE may consist of a cluster of computers; if some of these machines go down for maintenance, the aggregate server capacity decreases temporarily. In other cases, a surge in client demand may temporarily overload the closest front-end server. In both cases, directing some clients

to an alternate front-end server is important for preventing degradation in performance. A slight increase in round-trip time for some clients is preferable to all clients experiencing slower downloads due to congestion.

To estimate the fraction of requests shifted by the load balancer, we identify front-end servers that handle a *lower* fraction of requests than what is suggested by the latency map. The latency map indicates that front-end server i should handle a fraction r_{1i} of the requests for the clients at time 1. In reality, the server handles RPD_{1i}/RPD_1.

$$LoadBalance_1 = \sum_i \left[r_{1i} - \frac{RPD_{1i}}{RPD_1} \right]^+$$

where $[]^+$ indicates that the sum only includes the positive values, with the target request load in excess of the actual load. Similarly, we define the fraction of queries load balanced at time 2 as $LoadBalance_2$.

If much more requests are load balanced on the second day, then more requests are directed to alternative FEs that are further away, leading to higher round-trip times. Thus, we use the difference of the load balancer metric to capture more load balancing traffic at time 2:

$$\Delta LoadBal = LoadBalance_2 - LoadBalance_1$$

We expect the load-balancing policy to routinely trigger some small shifts in traffic.

D. Inter-domain Routing Changes

Next, our analysis focuses on events where the RTT jumps significantly for specific FEs. These increases in round-trip time could be caused by routing changes, or by congestion along the paths to and from the client. Since the CDN does not have direct visibility into congestion outside its own network, we correlate the RTT increases only with the routing changes visible to the CDN—changes of the ingress router where client traffic enters the CDN network, and changes of the egress router and the AS path used to reach the client.

Recall that the latency metric ΔLat can be broken down to the sum of latency metrics at individual FEs (i.e., ΔLat_i). We focus our attention on the FE with the highest value of ΔLat_i, because the latency change for requests to this FE has the most impact on the latency increase seen by the clients. Then, we define metrics to capture what fraction of the traffic destined to this FE experiences a visible routing change. Focusing on the front-end server i with the highest latency increase, we consider where the traffic enters the network. Given all the traffic from the client region to the front-end server, we can compute the fractions f_{1j} and f_{2j} entering at ingress router j at time 1 and time 2, respectively. Note that we compute these fractions from the "bytes-in" statistics from the Netflow data, since the front-end server cannot differentiate the requests per day (RPD) by which ingress router carried the traffic.

To quantify how traffic shifted to different ingress routers, we compute:

$$\Delta Ingress = \sum_j |f_{2j} - f_{1j}|/2$$

Note that the difference between the fractions is divided by two, to avoid double counting traffic that shifts away from one ingress router and toward another. Similarly, we define a metric to measure the fraction of traffic to a FE that switches to a different egress router or AS path. Suppose the fraction of traffic to (egress router, AS path) k is g_{1k} at time 1 and g_{2k} at time 2. Then,

$$\Delta EgressASPath = \sum_k |g_{2k} - g_{1k}|/2$$

similar to the equation for analyzing the ingress routers. These metrics allow us to correlate large increases in latency to server i with observable routing changes. Note that the analysis can only establish a correlation between latency increases and routing changes, rather than definitively "blaming" the routing change for the higher delay, since the performance measurements cannot distinguish RTT by which ingress or egress router carried the traffic.

IV. Distribution of Latency Changes

In the rest of the paper, we apply our tool to measurement data from Google's CDN. The BGP and Netflow data are collected and joined on a 15-minute timescale; the performance data is collected daily, and joined with the routing and traffic data to form a joint data set for each day in June 2010. For our analysis, we focus on the large latency increases which last for a long time and affect a large number of clients. We pick daily changes as the timescale, because the measurement data we get is aggregated daily. We group clients by "region," combining all IP addresses with the same origin AS *and* located in the same country. In this section, we describe how we preprocess the data, and characterize the distribution of daily increases in latency to identify the most significant events which last for days. We also determine the threshold for the large latency increases we study.

As our datasets are proprietary, we are not able to reveal the exact number of regions or events, and instead report percentages in our tables and graphs; we believe percentages are more meaningful, since the exact number of events and regions naturally differ from one CDN to another. In addition, the granularity of the data, both spatially (i.e., by region) and temporally (i.e., by day) are beyond our control; these choices are not fundamental to our methodology, which could easily be applied to finer-grain measurement data.

A. Aggregating Measurements by Region

Our joint dataset has traffic and performance data at the level of BGP prefixes, leading to approximately 250K groups of clients to consider. Many of these prefixes generate very little traffic, making it difficult to distinguish meaningful changes in latency from statistical noise. In addition, CDN administrators understandably prefer to have more concise summaries of significant latency changes that affect many clients, rather than reports for hundreds of thousands of prefixes.

Combining prefixes with the same origin AS seems like a natural way to aggregate the data, because many routing and traffic changes take place at the AS level. Yet, some ASes

are quite large in their own right, spanning multiple countries. We combine prefixes that share the same country and origin AS (which we define as a *region*), for our analysis. From the performance measurements, we know the country for each /24 prefix, allowing us to identify the country (or set of countries) associated with each BGP prefix. A prefix spanning multiple countries could have large variations in average RTT simply due to differences in the locations of the active clients. As such, we filter the small number of BGP prefixes spanning multiple countries. This filters approximately 2K prefixes, which contribute 3.2% of client requests and 3.3% of the traffic volume.

After aggregating clients by region, some regions still contribute very little traffic. For each region, we calculate the minimum number of requests per day (RPD) over the month of June 2010. We choose a threshold for the minimum RPD to filter the regions with very low client demand. This process improves statistical accuracy, because it makes sure that we have enough samples of requests for the regions we study. This also helps focus our attention on regions with many clients, and reduce the volume of the measurement data we analyze. This process helps us to exclude the regions ranging at the long tail in traffic distribusion. After this preprocessing step, our experiments still cover 94% of the traffic, which include 15% of the regions. We also ensure that these regions cover all the major geographical areas globally and all the major ASes.

Hence, for the rest of our analysis, we focus on clients aggregated by region (i.e., by country and origin AS), and regions generating a significant number of requests per day. Note that our analysis methodology could be applied equally well to alternate ways of aggregating the clients and filtering the data.

The measurement results we present in the following sections cover all the days in the month of June 2010. The data represents the global traffic we receive at Google, and we ensure that all the major geographical areas and large ASes are covered.

B. Identifying Large Latency Increases

To gain an initial understanding of latency changes, we first characterize the differences in latency from one day to the next throughout the month of June 2010, across all the client regions we selected. We consider both the *absolute* changes (i.e., $RTT_2 - RTT_1$) and the *relative* change (i.e., $(RTT_2 - RTT_1)/RTT_1$), as shown in Figures IV-A(a) and IV-A(b), respectively. The graphs plot only the *increases* in latency, because the distributions of daily increases and decreases are symmetric.

The two graphs are plotted as complementary cumulative distributions, with a logarithmic scale on both axes, to highlight the large outliers. Figure IV-A(a) shows that latency increases less than 10msec for 79.4% of the time. Yet, nearly 1% of the latency increases exceed 100 msec, and every so often latency increases by more than one second. Figure IV-A(b) shows that the RTT increases by less than 10% in 80.0% of cases. Yet, the daily RTT at least doubles (i.e., a relative increase of 1 or more) for 0.45% of the time, and we see occasional increases by a factor of ten.

TABLE III
EVENTS WITH A LARGE DAILY RTT INCREASE

Category	% Events
Absolute RTT Increase $\geq$ 100 ms	76.9%
Relative RTT Increase $\geq$ 1	35.6%
Total large events	100%

TABLE IV
CAUSES OF LARGE LATENCY INCREASES (WHERE LATENCY MORE THAN DOUBLES, OR INCREASES BY MORE THAN 100 MSEC), RELATIVE TO PREVIOUS DAY. NOTE THAT NEARLY 9% OF EVENTS INVOLVE BOTH FE LATENCY INCREASES AND FE SERVER CHANGES.

Category	% Events
FE latency increase	*73.9%*
Ingress router	10.3%
(Egress, AS Path)	14.5%
Both	17.4%
Unknown	31.5%
FE server change	*34.7%*
Latency map	14.2%
Load balancing	2.9%
Both	9.3%
Unknown	8.4%
Total	*100.0%*

We define an *event* to be a daily RTT increase over a threshold for a specific region. Table IV-B summarizes the events we selected to characterize the latency increase. We choose the threshold of absolute RTT increase as 100 ms and the threshold of relative RTT increase as 1, leading to a combined list of hundreds of events corresponding to the most significant increases in latency: with 76.9% of the events over the absolute RTT increase threshold; 35.6% of the events over the relative RTT increase threshold; and 12.5% of the events over both thresholds.

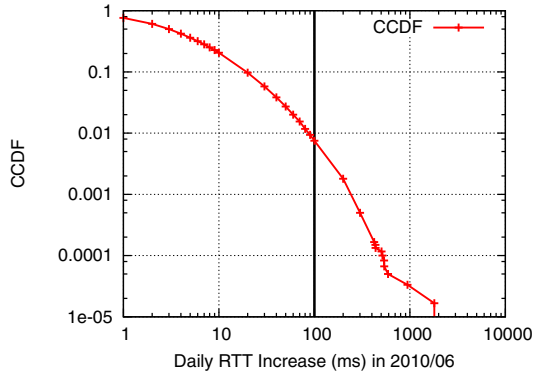
V. LATLONG DIAGNOSIS OF LATENCY INCREASES

In this section, we apply our tool to study the events of large latency increases, which are identified in the previous section. We first classify them into FE changes and latency increases at individual FEs. Then, we further classify the events of FE changes according to the causes of the latency map and load balancing; classify the events of FE latency increases according to the causes of inter-domain routing changes.

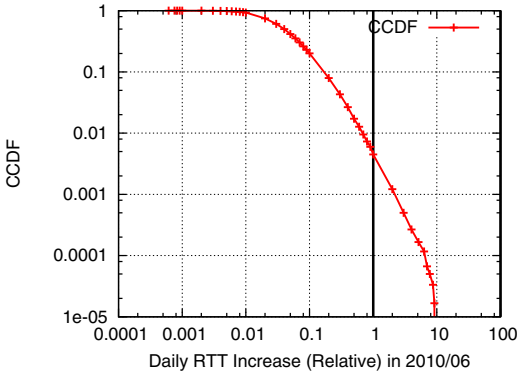
Our high-level results in this section are summarized in Table V. Nearly three-quarters of these events were explained (at least in part) by a large increase in latency to reach an existing front-end server. These latency increases often coincided with a change in the ingress router or egress router (or both!); still, many had no visible interdomain routing change and were presumably caused by BGP routing changes on the forward path or by congestion or intradomain routing changes. Around one-third of the events involved a significant shift of client traffic to different front-end servers, often due to load-balancing decisions or changes in CDN's own view of the closest server. Nearly 9% of events involved both an "FE latency increase" *and* an "FE server change," which is why they sum to more than 100%.

A. FE Change vs. Latency Increase

Applying our tool to each event identified in the last section, we see that large increases in latency to reach existing servers (i.e., ΔLat) are responsible for more than two-thirds of the



(a) Absolute RTT increase



(b) Relative RTT increase

Fig. 3. Distribution of daily RTT increase

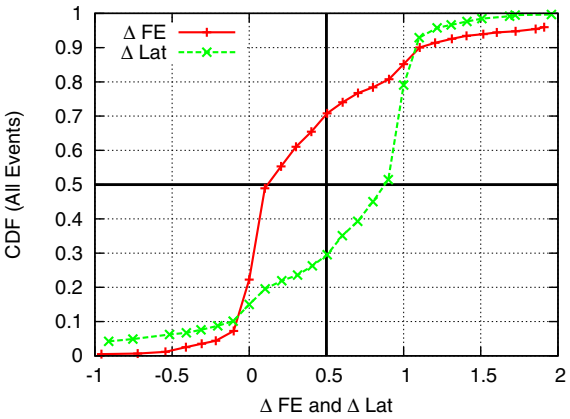


Fig. 4. ΔFE and ΔLat for large events.

events with a large increase in round-trip time. To identify the cause of latency increases, we first show the CDF of ΔFE (traffic shift) and ΔLat (latency increase) for the events we study in Figure 4. The distributions are a reflection of each other (on both the x and y axes), because ΔFE and ΔLat sum to 1 for each event.

The graph shows that about half of the events have ΔFE below 0.1, implying that shifts in traffic from one FE to another are not the major cause of large-latency events. Still, traffic shifts are responsible for *some* of the latency increases—one event has a ΔFE of 5.83! (Note that we do not show the very few points with extreme ΔFE or ΔLat values, so we can illustrate the majority of the distribution more clearly in the graph). In comparison, ΔLat is often fairly high—in fact, more than 70% of these events have a ΔLat higher than 0.5.

To classify these events, we apply a threshold to both distributions and identify whether ΔFE or ΔLat (or both) exceeds the threshold. Table V-A summarizes the results for thresholds 0.3, 0.4, and 0.5. These results show that, for a range of thresholds, around two-thirds of the events are explained primarily by an increase in latency between the clients and the FEs. For example, using a threshold of 0.4 for both distributions, 65% of events have a large ΔLat and another 9% of events have large values for both metrics, resulting in nearly three-quarters of the events caused (in large part) by increases in RTTs to select front-end servers. In the

TABLE V
EVENTS CLASSIFIED BY ΔLat AND ΔFE

	Threshold		
	0.3	**0.4**	**0.5**
ΔLat	61%	65%	71%
ΔFE	23%	26%	29%
Both	16%	9%	0%

rest of the paper, we apply a threshold of 0.4 to distinguish events into the three categories in Table V-A. This is because the threshold of 0.5 separates the two categories apart; the threshold of 0.3 (where one factor contributes to 30% of the latency increases) is not as significant as 0.4.

B. Normal Front-End Changes

To understand the normal distribution of latency-map changes, we calculate $\Delta LatMap$ for *all* of the regions—whether or not they experience a large increase in latency—on two consecutive days in June 2010. Figure 5 shows the results. For 76.9% of the regions, less than 10% of the requests change FEs because of changes to the latency map. For 85.7% of regions, less than 30% of traffic shifts to different front-end servers. Less than 10% of the regions see more than half of the requests changing front-end servers. Often, these changes involve shifts to another front-end server in a nearby geographic region.

However, note that the distribution of $\Delta LatMap$ has a long tail, with some regions having 80% to 90% of the requests shifting FEs. For these regions, changes in the measured latency lead to changes in the latency map which, in turn, lead to shifts in traffic to different front-end servers. These outliers are not necessarily a problem, though, since the FEs on the second day may be very close to the FEs on the first day. To understand the impact of these traffic shifts, we need to consider the resulting latency experienced by the clients.

Figure 5 also shows the resulting distribution of $\Delta FEDist$ (i.e., the actual FE changes) for all client regions for one pair of consecutive days in June 2010. As expected, the distribution matches relatively closely with the distribution for $\Delta LatMap$, though some significant differences exist. Sometimes the traffic shifts even though the latency map does not change. This is evident in the lower left part of the graph, where 40% of the client regions see little or no change to the

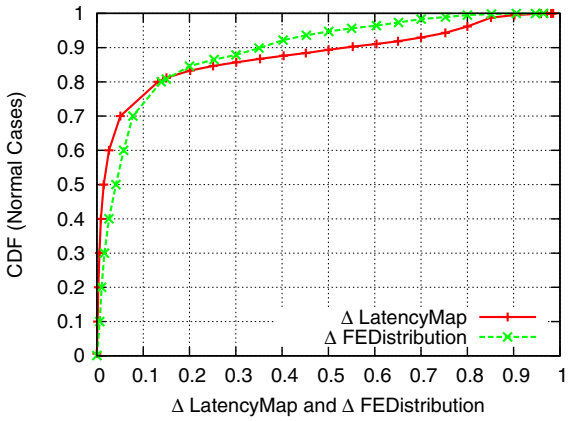


Fig. 5. Distribution of $\Delta LatMap$ and $\Delta FEDist$ across all client regions for one day in June 2010.

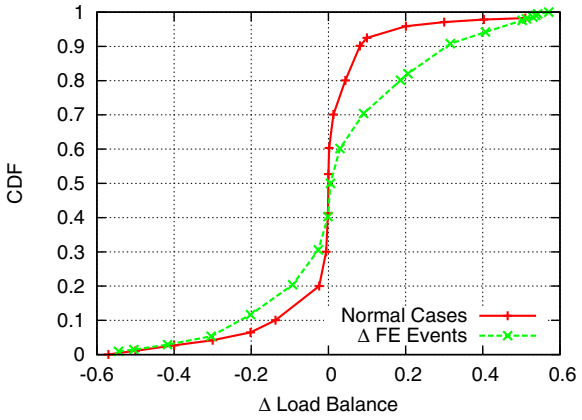


Fig. 6. Distribution of $\Delta LoadBalance$ for normal cases and events $\Delta FE \geq 0.4$.

latency map, but more than half of the regions experience as much as a 5% shift in traffic.

We expect the load-balancing policy to routinely trigger some small shifts in traffic. Figure 6 plots the distribution of $\Delta LoadBal$ for all client regions for a single day in June 2010, as shown in the "Normal Cases" curve. As expected, around 30% of the client regions are directed to the closest front-end server, as indicated by the clustering of the distribution around $\Delta LoadBal = 0$. In the next subsection, we show that the large latency events coincide with larger shifts in traffic, as illustrated by the "ΔFE Events" curve in Figure 6.

C. Front-End Changes During Events

To understand the influence of traffic shifts during the events, we analyze the large-latency events where front-end changes are a significant contributor to the increase in latency (i.e., $\Delta FE \geq 0.4$); 35% of the events fall into this category, as shown earlier in Table V-A. Figure 7 plots the distributions of $\Delta LatMap$ and $\Delta FEDist$ for these events. For these events, the FE distribution still mostly agrees with the latency map. Compared with the curves in Figure 5, the events which experienced large latency increases have a stronger correlation with FE changes. According to the latency map, only 14% of events have fewer than 10% of requests changing FEs; 46% of the events have more than half of queries shifting FEs. Note that FE changes (i.e., in nearby geographical locations) do

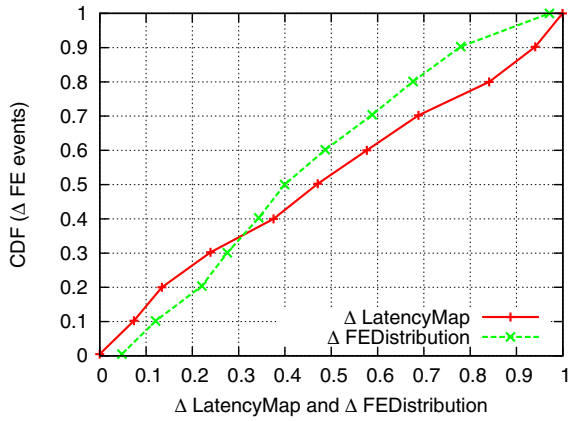


Fig. 7. Distribution of $\Delta LatencyMap$ and $\Delta FEDistribution$ for events $\Delta FE \geq 0.4$.

not necessarily lead to large latency increases, and may even improve user-perceived throughput by avoiding busy servers. That said, these FE changes can cause increases in round-trip time, so we need to understand how and why they happen.

We then calculate the $\Delta LoadBal$, the difference of fraction of traffic directed by the load balancer from one day to the next. Figure 6 shows the distribution of $\Delta LoadBal$ for these events and for all client regions. As illustrated in the figure, 92.5% of the normal cases have less than 10% of requests shifted away from the closest front-end server. In contrast, for the ΔFE events, 27.7% of the events have a $\Delta LoadBal$ value greater than 10%; more than 9% of the events have a $\Delta LoadBal$ in excess of 0.3, suggesting that the load-balancing policy is responsible for many of the large increases in latency.

Based on the $\Delta LatMap$ and $\Delta LoadBal$ metrics, we classify the events into four categories: (i) correlated only with latency map changes, (ii) correlated only with load balancing changes, (iii) correlated with both latency-map changes and load balancing; and (iv) unknown. We choose the 85th-percentile and 90th-percentile in the distribution for the normal cases as the thresholds for $\Delta LatMap$ and $\Delta LoadBal$. Table V-C summarizes the results: 26.7% of the events are correlated with both changes to the latency map and load balancing; 40.8% of the events only with changes in the latency map; 8.3% of the events only with load balancing; and 24.3% of the events fall into the unknown category. The table also shows results for the 90th-percentile thresholds.

Note that in the "unknown" category, although the fraction of traffic shifting FEs is low, this does not mean that the FE change is not responsible for the latency increases. This is because: what matters is the latency difference between the FEs, not only the fraction of traffic shifting FEs. For these events in the unknown category, we still need to analyze how much the latency differs between the FEs from one day to the next; we suspect that, while the fraction of traffic shifting is small, the absolute increase in latency may be high. Completing this analysis is part of our ongoing work.

D. Inter-domain Routing Changes

In this subsection, we study the events where the round-trip time increases to existing front-end servers. We characterize

TABLE VI
CLASSIFICATION OF EVENTS WITH $\Delta FE \geq 0.4$

Threshold (Percentile)	**(0.27, 0.06) 85th**	**(0.53, 0.08) 90th**
Latency Map	40.8%	23.8%
Load Balancing	8.3%	12.6%
Both	26.7%	18.4%
Unknown	24.3%	45.1%

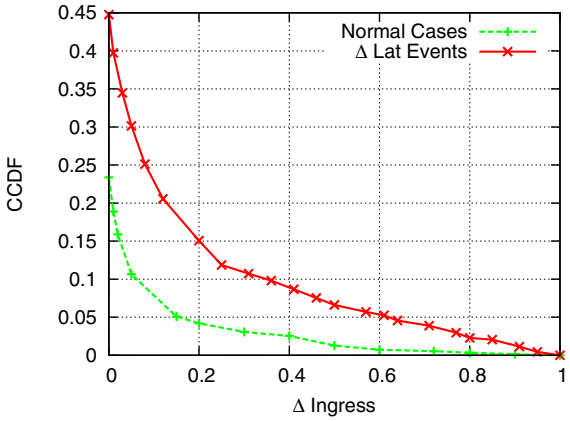


Fig. 8. Ingress router shifts ($\Delta Ingress$).

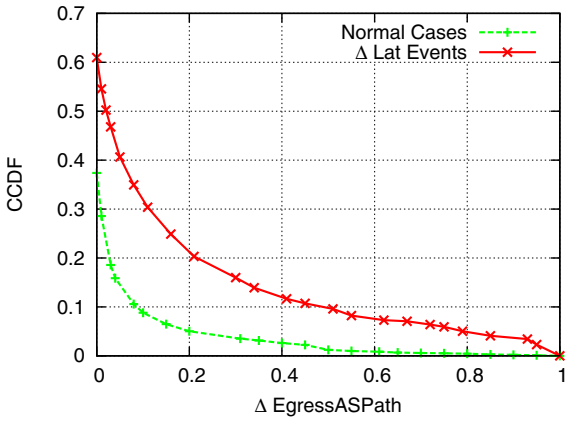


Fig. 9. Egress router and AS path shifts ($\Delta EgressASpath$).

TABLE VII
CLASSIFICATION OF EVENTS WITH $\Delta Lat \geq 0.4$

Thresholds (Percentile)	**(0.025, 0.05) 85th**	**(0.06, 0.09) 90th**
Ingress	13.9%	12.6%
Egress/AS-path	19.6%	17.4%
Both	23.7%	17.6%
Unknown	42.7%	52.5%

the events based on these metrics, and classify events based on changes to the ingress router, the egress router and AS path, or both.

For better insight into whether routing changes are responsible for latency increases, we first consider the prevalence of routing changes for *all* client regions—when latency does *not* necessarily increase significantly—for a pair of consecutive days in June 2010. Figure 8 shows the CCDF, with the y-axis cropped at 0.45, to highlight the tail of the distribution where clients experience a large shift in ingress routers. Significant routing changes are relatively rare for the "Normal Cases." In fact, 76.6% of the client regions experience *no* change in the distribution of traffic across ingress routers. Less than 7% of the regions experience a shift of more than 10%. As such, we see that shifts in where traffic enters Google's CDN network do not occur often, and usually affect a relatively small fraction of the traffic.

However, large shifts in ingress routers are more common for the events where the round-trip time to a front-end server increases significantly (i.e., $\Delta Lat \geq 0.4$), as shown by the "ΔLat Events" curve in Figure 8. The events we study have a much stronger correlation with changes in the ingress routers, compared with the normal cases. Though 55% of these events do not experience *any* change in ingress routers, 22.2% of events see more than a 10% shift, and 6.7% of the events see more than *half* of the traffic shifting ingress routers.

Similarly, we calculate $\Delta EgressASPath$ for both the normal cases and the ΔLat events, as illustrated in Figure 9. Compared with ingress changes, we see more egress and AS path changes, in part because we can distinguish routing changes at a finer level of detail since we see the AS path. For the normal cases, 63% of the client regions see no change in the egress router or the AS path; 91% see less than 10% of the traffic shifting egress router or AS path. In comparison, for the "ΔLat Events," only 39% of the events see no changes in the egress routers and AS paths; 32% of the events see more than 10% of the traffic changing egress router or AS path, and 10% of the events see more than *half* of the traffic shifting egress routers and/or AS paths.

Based on both of the routing indicators, we classify the events into four categories: (i) correlated only with ingress router changes, (ii) correlated only with changes in the egress router and AS path, (iii) correlated with both ingress changes and egress/AS-path changes, and (iv) unknown. To identify significant shifts, we look to the distributions for "Normal Cases" and consider the 85th and 95th percentiles for shifts in both $\Delta Ingress$ and $\Delta EgressASPath$. Table V-D summarizes the results. Based on the 85th-percentile thresholds, 23.7% of the events are associated with large shifts in both the ingress routers and the egress/AS-path; 13.9% of the events are associated with ingress-router shifts; 19.6% of the events are associated with shifts in the egress router and AS path; and 42.7% of the events fall into the unknown category. We also show results using the 90th-percentile thresholds.

Note that around half of the events fall into the unknown category, where we could not correlate latency increases with large, visible changes to interdomain routing. Potential explanations include AS-level routing changes on the forward path (from the client to the front-end server) that do not affect where traffic enters Google's CDN network. Intradomain routing changes in individual ASes could also cause increases in round-trip time without changing the ingress router, egress router, or AS path seen by the CDN. Finally, congestion along either the forward or reverse path could be responsible. These results suggest that CDNs should supplement BGP and traffic data with finer-grain measurements of the IP-level forwarding path (e.g., using traceroute and reverse traceroute [5]) both for better accuracy in diagnosing latency increases and to drive new BGP path-selection techniques that make routing decisions based on direct observations of performance.

VI. Case Studies

For a better understanding of large latency increases, we explore several events in greater detail. These case studies illustrate the general challenges CDNs face in minimizing wide-area latency and point to directions for future work. Although many of these problems are known already, our case studies highlight that these issues arise in practice and are responsible for very large increases in latency affecting real users.

A. Latency-Map Inaccuracies

During one day in June 2010, an ISP in the United States saw the average round-trip time increase by 111 msec. Our analysis shows that the RTT increased because of a shift of traffic to different front-end servers; in particular, ΔFE was 1.01. These shifts were triggered primarily by a change in the latency map; in particular, $\Delta LatMap$ was 0.90. Looking at the latency map in more detail revealed the reason for the change. On the first day, 78% of client requests were directed to front-end servers in the United States, and 22% were directed to servers in Europe. In contrast, on the second day, all requests were directed to front-end servers in Europe. Hence, the average latency increased because the clients were directed to servers that were further away. The situation was temporary, and the clients were soon directed to closer front-end servers.

This case study points to the challenges of identifying the closest servers and using DNS to direct clients to servers—topics explored by several other research studies [6], [4], [7], [8], [9]. Clients do not necessarily reside near their local DNS servers, especially with the increasing use of services like GoogleDNS and OpenDNS. Similarly, client IP addresses do not necessarily fall in the same IP prefix as their local DNS server. Further, DNS caching causes the local DNS server to return the same IP address to many clients over a period of time. All of these limitations of DNS make it difficult for a CDN to exert fine-grain control over server selection. Recent work at the IETF proposes extensions to DNS so requests from local DNS servers include the client's IP address [10], which should go a long way toward addressing this problem. Still, further research on efficient measurement techniques and efficient, fine-grain control over server selection would be very useful.

B. Flash Crowd Leads to Load Balancing to Distant Front-End Servers

As another example, we saw the average round-trip time double for an ISP in Malaysia. The RTT increase was caused by a traffic shift to different front-end servers; in particular, ΔFE was 0.979. To understand why, we looked at the metrics for front-end server changes. First, we noticed that $\Delta LatMap$ was 0.005, suggesting that changes in the latency map were not responsible. Second, we observed that $\Delta FEDist = 0.34$ and $\Delta LoadBal = 0.323$, suggesting that load balancing was responsible for the shift in traffic. Looking at the client request rate, we noticed that the requests per day jumped significantly from the first day to the second; in particular, $RPD_2/RPD_1 = 2.5$. On the first day, all requests were served as front-end servers close to the clients; however, on the second day, 40% of requests were directed to alternate front-end servers that were further way. This led to a large increase in the average round-trip time for the whole region.

This case study points to a general limitation of relying on round-trip times as a measure of client performance. If, on the second day, Google's CDN had directed all client requests to the closest front-end server, the user-perceived performance would likely have been *worse*. Sending more requests to an already-overloaded server would lead to slow downloads for a very large number of clients. Directing some requests to another server—even one that is further away—can result in higher throughput for the clients, including the clients using the remote front-end server. Understanding these effects requires more detailed measurements of download performance, and accurate ways to predict the impact of alternate load-balancing strategies of client performance. We believe these are exciting avenues for future work, to enable CDNs to handle flash crowds and other shifts in user demand as effectively as possible.

C. Shift to Ingress Router Further Away from the Front-End Server

On day in June 2010, an ISP in Iran experienced an increase of 387 msec in the average RTT. We first determined that the RTT was mainly caused by a large increase in latency to reach a particular front-end server in western Europe. This front-end server handled 65% of the requests on both days. However, ΔLat_i for this server was 0.73, meaning 73% of the increase in RTT was caused by an increase in latency to reach this front-end server. Looking at the routing changes, we saw a $\Delta Ingress$ of 0.38. Analyzing the traffic by ingress router, we found that, on the first day, all of the traffic to this front-end server entered the CDN's network at a nearby ingress router in western Europe. However, on the second day, nearly 40% of the traffic entered at different locations that were further away—21% in eastern Europe and 17% of traffic in the United States. Thus, the increase in RTT was likely caused by extra latency between the ingress router and the front-end server, and perhaps also by changes in latency for the clients to reach these ingress routers.

This case study points to a larger difficulty in controlling *inbound* traffic using BGP. To balance load over the ingress routers, and generally reduce latency, a large AS typically announces its prefixes at many locations. This allows other ASes to select interdomain routes with short AS paths and nearby peering locations. However, an AS has relatively little control over whether other ASes can (and do) make good decisions. In some cases, a CDN may be able to use the Multiple Exit Discriminator (MED) attribute in BGP to control how individual neighbor ASes direct traffic, or perform selective AS prepending or selective prefix announcements to make some entry points more attractive than others. Still, this is an area that is ripe for future research, to give CDNs more control over how clients reach their services.

D. Shorter AS Paths Not Always Better

On another day in June 2010, an ISP in Mauritius experienced a 113 msec increase in the average round-trip time. On

both days, more than half of the client requests were handled by a front-end server in Asia—60% on the first day and 74% on the second day. However, on the second day, the latency to reach this front-end server increased substantially. Looking at the routing data, we see that traffic shifted to a different egress router and AS path. On the first day, 56% of the traffic left Google's CDN's network in Asia. On the second day, this number dropped to 10%, and nearly two-thirds of the traffic left the network in Europe over a *shorter* AS path. Presumably, upon learning a BGP route with a shorter AS path, the routers preferred this route over the "longer" path through Asia. However, AS-path length is (at best) loosely correlated with round-trip time, and in this case the "shorter" path had a much higher latency.

This case study points to a larger problem with today's interdomain routing system—routing decisions do not consider performance. The BGP decision process uses AS-path length as a (very) crude measure of performance, rather than considering measurements of actual performance along the end-to-end paths. Future work could explore lightweight techniques for measuring the performance along different interdomain paths, including the paths not currently selected for carrying traffic to clients. For example, recent work [11] introduces a "route injection" mechanism for sampling the performance on alternative paths. Once path performance is known, CDNs can optimize interdomain path selection based on performance, load, and cost. However, large CDNs with their own backbone network introduce two interesting twists on the problem of intelligent route control. First, the CDN selects interdomain routes at *multiple* egress points, rather than a single location. Second, the CDN can *jointly* control server selection and route selection for much greater flexibility in directing traffic.

VII. Future Research Directions

In this section, we briefly discuss several natural directions for future work on diagnosing wide-area latency increases for CDNs.

Direct extensions of our measurement study: First, we plan to extend our design in Section III to distinguish between routing changes that affect the egress router from those that only change the AS path. Second, as discussed at the end of Section V-C, we plan to further explore the unexplained shifts in traffic from one front-end server to another. We suspect that some of these shifts are caused by a relatively small fraction of traffic shifting to a much further away front-end server. To analyze this further, we plan to incorporate the RTT differences between front-end servers as part of our metrics for studying FE changes. Third, our case studies in Section VI required manual exploration, after automatically computing the various metrics. We plan to conduct more case studies and automate the analysis to generate reports for the network operators.

More accurate diagnosis: First, we plan to work with the groups that collect the measurement data to provide the data on a smaller timescale (to enable finer-grain analysis) and in real time (to enable real-time analysis). Second, we plan to explore better ways to track the performance data (including RTT and RPD) separately for each ingress router and egress/AS-path. Currently, the choice of ingress and egress routers are not visible to the front-end servers, where the performance data are collected. Third, we will explore techniques for correlating across latency increases affecting multiple customer regions. For example, correlating across interdomain routing changes that affect the AS paths for multiple client prefixes may enable us to better identify the root cause [12].

Incorporating additional data sets: We plan to investigate techniques for improving the visibility of the routing and performance changes from outside the CDN network. For example, active measurements—such as performance probes and traceroute (including both forward and reverse traceroute [13])—would help explain the "unknown" category for the ΔLat events, which we could not correlate with visible routing changes. In addition, measurements from the front-end servers could help estimate the performance of alternate paths, to drive changes to the CDN's routing decisions to avoid interdomain paths offering poor performance.

VIII. Related Work

CDNs have been widely deployed to serve Web content. In these systems, clients are directed to different servers to reduce latency and balance load. Our classification reveals the main causes of high latency between the clients and the servers.

An early work in [6] studied the effectiveness of DNS redirection and URL rewriting in improving client performance. This work characterizes the size and the number of the web objects CDNs served, the number of distinct IP addresses used in DNS redirection, and content download time, and compared the performance for a number of CDN networks. Recent work in [14] evaluated the performance of two large-scale CDNs—Akamai and LimeLight. Instead of measuring CDNs from end hosts, we design and evaluate techniques for a CDN to diagnose wide-area latency problems, using readily-available traffic, performance, and routing data.

WhyHigh [3] combines active measurements with routing and traffic data to identify causes of persistent performance problems for some CDN clients. For example, WhyHigh identifies configuration problems and side-effects of traffic engineering that lead some clients to much higher latency than others in the same region. In contrast, our work focuses on detecting and diagnosing large *changes* in performance over time, and also considers several causes of traffic shifts from one front-end server to another. The *dynamics* of latency increases caused by the changes in FE server selection, load balancing, and inter-domain routing changes are not studies in the work of WhyHigh. WISE [15] predicts the effects of possible configuration and deployment changes in the CDN. Our work is complementary in that, instead of studying planned maintenance and operations, we study how to detect and diagnose unplanned increases in latency.

PlanetSeer [16] uses passive monitoring to detect network path anomalies in the wide-area, and correlates active probes to characterize these anomalies (temporal vs. persistent, loops, routing changes). The focus of our work is different in that, instead of characterizing the end-to-end *effects* of performance anomalies, we study how to classify them according to the

causes. Recent work [17] measures wide-area performance for CoralCDN using kernel-level TCP statistics, and identified causes of performance problems such as server-limits and the congestion window. In comparison, we focus on the causes of performance problems at the IP layer, related to the CDN network design and Internet routing.

Note that management of wide-area performance of CDN services is a relatively new topic, and a heavily commercial topic, so not many published papers are available on how CDN management is done today.

IX. Conclusion

The Internet is increasingly a platform for users to access online services hosted on servers distributed throughout the world. Today, ensuring good user-perceived performance is a challenging task for the operators of large Content Distribution Networks (CDNs). In this paper, we presented the system design for automatically classifying large changes in wide-area latency for CDNs, and the results from applying our methodology to traffic, routing, and performance data from Google. Our techniques enable network operators to learn quickly about significant changes in user-perceived performance for accessing their services, and adjust their routing and server-selection policies to alleviate the problem.

Using only measurement data readily available to the CDN, we can automatically trace latency changes to shifts in traffic to different front-end servers (due to load-balancing policies or changes in the CDN's own view of the closest server) and changes in the interdomain paths (to and from the clients). Our analysis and case studies suggest exciting avenues for future research to make the Internet a better platform for accessing and managing online services.

X. Acknowledgments

We thank Roshan Baliga, Andre Broido and Mukarram Tariq for their valuable feedback in the early stages of this work, as well as Bo Fu for iterating on the implementation details of the prototype. Special thanks to Ankur Jain for his insightful comments on FE changes. We are also grateful to Murtaza Motiwala, Srinivas Narayana, Vytautas Valancius, the anonymous reviewers and the editors for their comments and suggestions.

References

[1] M. Szymaniak, D. Presotto, G. Pierre, and M. V. Steen, "Practical large-scale latency estimation," *Computer Networks*, 2008.

[2] Cisco NetFlow, http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

[3] R. Krishnan, H. V. Madhyastha, S. Srinivasan, and S. Jain, "Moving beyond end-to-end path information to optimize CDN performance," in *Proc. 2009 Internet Measurement Conference*.

[4] Z. M. Mao, C. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang, "A precise and efficient evaluation of the proximity between web clients and their local DNS servers," in *Proc. 2002 USENIX Annual Technical Conference*.

[5] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. van Wesep, T. Anderson, and A. Krishnamurthy, "Reverse traceroute," in *Proc. 2010 Networked Systems Design and Implementation*.

[6] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proc. 2001 Internet Measurement Workshop*.

[7] A. Shaikh, R. Tewari, and M. Agarwal, "On the effectiveness of DNS-based server selection," in *Proc. 2002 IEEE INFOCOM*.

[8] J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan, "On the responsiveness of DNS-based network control," in *Proc. 2004 Internet Measurement Conference*.

[9] B. Ager, W. Muehlbauer, G. Smaragdakis, and S. Uhlig, "Comparing DNS resolvers in the wild," in *Proc. 2010 Internet Measurement Conference*.

[10] C. Contavalli, W. van der Gaast, S. Leach, and D. Rodden, "Client IP information in DNS requests," May 2010, Internet Draft, draft-vandergaast-edns-client-ip-01.

[11] Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian, "Optimizing cost and performance in online service provider networks," in *Proc. 2010 Networked Systems Design and Implementation*.

[12] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating Internet routing instabilities," in *Proc. 2004 ACM SIGCOMM*.

[13] E. Katz-Bassett, H. Madhyastha, V. K. Adhikar, C. Scott, J. Sherry, P. V. Wesep, T. Anderson, and A. Krishnamurthy, "Reverse traceroute," in *Proc. 2010 USENIX/ACM NSDI*.

[14] C. Huang, A. Wang, J. Li, and K. W. Ross, "Measuring and evaluating large-scale CDNs," Microsoft Research Technical Report MSR-TR-2008-106, 2008.

[15] M. B. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, "Answering 'what-if' deployment and configuration questions with WISE," in *Proc. 2008 ACM SIGCOMM*.

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

[17] P. Sun, M. Yu, M. Freedman, and J. Rexford, "Identifying performance bottlenecks in CDNs through TCP-level monitoring," *2011 SIGCOMM Workshop on Measurements up the Stack*.

Yaping Zhu is a software engineer at Google Inc. Before joining Google, she was a research assistant in the Network Systems Group at Princeton University. She also worked at AT&T Labs Research and NEC Labs America on research and development of network management tools. Yaping Zhu received her BS in computer science from Peking University in 2005, and her MA and Ph.D. degrees in computer science from Princeton University in 2007 and 2011.

Benjamin Helsley is a software engineer at Google Inc. where he works on systems for network monitoring and configuration management. Prior to joining Google, Benjamin worked at MacDonald Dettwiler and Safe Software on algorithms for UAV cooperation and GIS data transformation. He received his BS in computer science from the University of British Columbia in 2008.

Jennifer Rexford is a Professor in the Computer Science department at Princeton University. From 1996-2004, she was a member of the Network Management and Performance department at AT&T Labs–Research. Jennifer is co-author of the book *Web Protocols and Practice* (Addison-Wesley, May 2001). She served as the chair of ACM SIGCOMM from 2003 to 2007, and is a senior member of the IEEE. Jennifer received her BSE degree in electrical engineering from Princeton University in 1991, and her MSE and Ph.D. degrees in computer science and electrical engineering from the University of Michigan in 1993 and 1996, respectively. She was the 2004 winner of ACM's Grace Murray Hopper Award for outstanding young computer professional.

Aspi Siganporia is director of engineering at Google. From 1990-2006, he worked at various start-up and established networking companies in the Silicon Valley such as Netsys and Cisco Systems. Aspi received his BTech degree in electrical engineering from IIT Mumbai in 1982, and his MS degree in computer science from the University of Louisiana in 1986.

Sridhar Srinivasan is a software engineer at Google. Before joining Google, he was a research assistant in the Networking and Telecommunications Group at Georgia Tech. Sridhar received his B.E. in computer engineering from the Delhi Institute of Technology, M.S. in computer science from Iowa State University and Ph.D. in computer science from Georgia Tech.