Latency Imbalance Among Internet Load-Balanced Paths: A Cloud-Centric View

YIBO PI, University of Michigan, USA
SUGIH JAMIN, University of Michigan, USA
PETER DANZIG, Unaffiliated, USA
FENG QIAN, University of Minnesota - Twin Cities, USA

Load balancers choose among load-balanced paths to distribute traffic as if it makes no difference using one path or another. This work shows that the latency difference between load-balanced paths (called *latency imbalance*), previously deemed insignificant, is now prevalent from the perspective of the cloud and affects various latency-sensitive applications. In this work, we present the first large-scale measurement study of latency imbalance from a cloud-centric view. Using public cloud around the globe, we measure latency imbalance both between data centers (DCs) in the cloud and from the cloud to the public Internet. Our key findings include that 1) Amazon's and Alibaba's clouds together have latency difference between load-balanced paths larger than 20ms to 21% of public IPv4 addresses; 2) Google's secret in having lower latency imbalance than other clouds is to use its own well-balanced private WANs to transit traffic close to the destinations and that 3) latency imbalance is also prevalent between DCs in the cloud, where 8 pairs of DCs are found to have load-balanced paths with latency difference larger than 40ms.

We further evaluate the impact of latency imbalance on three applications (i.e., NTP, delay-based geolocation and VoIP) and propose potential solutions to improve application performance. Our experiments show that all three applications can benefit from considering latency imbalance, where the accuracy of delay-based geolocation can be greatly improved by simply changing how `ping` measures the minimum path latency.

CCS Concepts: • **Networks** → **Routing protocols**; **Network performance analysis**; **Network measurement**.

Additional Key Words and Phrases: Load-balanced paths; Latency imbalance; Cloud; Latency-sensitive

ACM Reference Format:
Yibo Pi, Sugih Jamin, Peter Danzig, and Feng Qian. 2020. Latency Imbalance Among Internet Load-Balanced Paths: A Cloud-Centric View. *Proc. ACM Meas. Anal. Comput. Syst.* 4, 2, Article 32 (June 2020), 29 pages. https://doi.org/10.1145/3392150

1 INTRODUCTION

Load balancers in the Internet can distribute traffic across multiple *load-balanced paths or LB paths* to better utilize network resources. This advantage of load balancing leads to its wide deployment in the Internet, with up to 90% of source-destination pairs reported as traversing a load balancer [1]. Load balancers commonly use equal-cost multi-path routing (ECMP) to find LB paths of equal cost (e.g., hop count) such that LB paths have similar performance [2]. However, as the cost of a path is not universally defined and equal cost does not necessarily equate similar performance, LB paths

Authors' addresses: Yibo Pi, University of Michigan, Ann Arbor, MI, USA, yibo@umich.edu; Sugih Jamin, University of Michigan, Ann Arbor, MI, USA, sugih@umich.edu; Peter Danzig, Unaffiliated, Palo Alto, CA, USA, pbdanzig@gmail.com; Feng Qian, University of Minnesota - Twin Cities, Minneapolis, MN, USA, fengqian@umn.edu.

2476-1249/2020/6-ART32 $15.00
https://doi.org/10.1145/3392150

could differ significantly in performance. Extensive efforts have been made to measure the topology of LB paths [1, 3, 4], but the performance difference between LB paths is still under-explored. In this work, we focus on the latency difference between LB paths and refer to this problem as *latency imbalance*.

Latency imbalance could affect application performance in various ways. First, latency imbalance implies that latency-sensitive applications, e.g., VoIP and cloud-based AR/VR, could have better quality of experience (QoE) by using lower-latency paths. Moreover, as 5G provides sub-millisecond latency at the last-mile ratio link [5], the Internet will dominate the end-to-end latency and latency imbalance will aggravate this situation. Second, latency imbalance makes the existing methods insufficient of measuring the minimum path latency, because they only consider the latency inflation in single paths [6], not the difference between paths. This affects geolocation applications or network coordinated systems relying on the accurate minimum path latency [7, 8]. Third, latency imbalance, if measured separately on the forward or return path, is an indicator to path asymmetry, affecting applications relying on symmetric paths to accurately estimate one-way delay, e.g., clock synchronization via networks [9, 10].

Latency imbalance was first studied by Augustin *et al.* in 2007 between 22K source-destination pairs, of which only 2% exhibit significant latency imbalance [1]. Low latency imbalance at that time was not enough to raise concerns. Although measurement tools, e.g., *Tokyo ping* [11], have been proposed to enforce probes to follow the same LB path to avoid instability of latency samples due to latency imbalance, existing measurement tools and applications do not consider latency imbalance by default. For instance, `ping` and `traceroute` are commonly used to measure latency in their default modes, allowing probes sent to the same destination to take different LB paths. In this work, we conduct the first large-scale measurement study of latency imbalance from a cloud-centric view, i.e., using data centers (DCs) as vantage points, and evaluate its impact on applications. In 2019, 12 years after the first study, we observed that 15.6% of 48M DC-destination pairs had significant latency imbalance, affecting various latency-sensitive applications.

To measure latency imbalance at the Internet scale, we need an efficient way of measuring it for each source-destination pair, which could have a large number of LB paths in between [4]. The major challenges are three-fold: 1) how to efficiently ensure that all LB paths are covered to calculate latency imbalance for a given source-destination pair? 2) how to efficiently obtain stable latency under transient network events for LB paths? 3) how to dissect latency imbalance to better understand the causes for latency imbalance? To address these challenges, our methodology uses three key strategies (§4.3). First, instead of enumerating LB paths that is inherently heavy in measurement overhead, we sample the latency of LB paths with sufficient sample size such that almost all LB paths are covered with high probability. Second, as we want to know how LB paths differ in latency, we use the range of latencies to measure the latency difference and obtain stable latencies only for LB paths that could potentially have the lowest and highest path latencies. Third, to better understand latency imbalance, we focus on measuring one-way latency imbalance occurring only on the forward paths that can be discovered with TTL-limited probes.

Using our methodology, we collect a global view of latency imbalance from the perspective of the cloud. The global view includes two parts: latency imbalance from DCs around the globe to about 3M /24 prefixes (§6) and latency imbalance between DCs in the cloud (§7). Overall, we find that latency imbalance is both significant and prevalent between DCs in the cloud and from the cloud to the public Internet. To better interpret the significance of latency imbalance, we determine the baseline of latency imbalance for three applications based on each application's sensitivity to latency imbalance (§5). We highlight our key findings below.

• Latency imbalance is prevalent from the perspective of DCs. Even the best DC has latency difference between LB paths larger than 5ms to about 20% of public IPv4 addresses and the latency

difference increases to 40ms to the same amount of addresses for the worst DC (§6). The prevalence of latency imbalance in the 1s of milliseconds could affect applications, e.g., Internet geolocation and clock synchronization via NTP, very sensitive to latency imbalance (§5). High latency imbalance in the tens of milliseconds mainly occurs on long-distance LB paths and thus would affect long-distance communications, e.g., international VoIP calls (§8).

• Latency imbalance differs significantly between cloud service providers: Google's DCs have consistently lower latency imbalance to public addresses than Amazon's and Alibaba's. Detailed path analysis tells us that Google uses private wide area networks (WANs) to route packets to an egress point close to the destinations and that its private long-distance paths are well balanced. Other cloud service providers, on the other hand, forward packets to the public Internet closer to traffic sources, which is less balanced than the providers' own networks (§6.2).

• The distribution of latency imbalance seen from most of the DCs is stable over months, which implies that latency imbalance is not due to temporary behaviors of the Internet (§6.3).

• Significant latency imbalance is found both between DCs in the same cloud (*intra-cloud*) and between DCs across different clouds (*inter-cloud*). Google's DCs (in premium network service tiers) have lower intra- and inter-cloud latency imbalance than those of other cloud providers due to using well-balanced paths in its own private WANs (§7).

• We further evaluate the impact of latency imbalance on three applications affected by latency imbalance in different ways. (1) Under latency imbalance, `ping` would overestimate the minimum path latency by > 5ms for 17% of addresses if measured from the cloud. Using the minimum path latency rather than the *ping-time* (the path latency measured by `ping`) could improve the accuracy of Internet geolocation by 40% for 20% of addresses for cloud-based applications, e.g., cloud data geolocation [12]. (2) Clock synchronization via NTP assumes symmetric paths to estimate one-way latency between the NTP client and server. Our experiments show that estimation error of the one-way latency can be reduced by as high as tens of milliseconds. (3) We simulate the visionary idea of using an overlay network, consisting of DCs around the globe as relays, for VoIP [13]. About 14% of long-distance VoIP calls between certain countries (e.g., US and CN) could have better call quality by using lower-latency paths.

In summary, our key contributions are as follows.

• We develop a methodology of efficiently measuring latency imbalance at the Internet scale. We carefully discuss the tradeoff between the accuracy and efficiency of our method and verify the accuracy of measured imbalance.

• We present a global view of latency imbalance from a cloud-centric view, where latency imbalance is found to be both significant and prevalent. We use path analysis to explain the difference between latency imbalance seen from different cloud providers and show that latency imbalance is stable over time.

• We evaluate the impact of latency imbalance on three applications and propose potential solutions to improve their performance.

• We make our tool and data publicly available at [14].

2 MOTIVATIONS AND BACKGROUND

To motivate our work, we first present the impact of latency imbalance and then introduce the background of Internet load-balancing.

2.1 Impact of Latency Imbalance

Latency imbalance affects application performance in various ways.

Latency-Sensitive Applications. Latency imbalance indicates that one path has less latency than the others. Using lower-latency paths would improve the QoE for applications sensitive to path latency. Moreover, in the era of 5G, the latency in cellular networks can be as low as 1ms and the latency bottleneck will be shifted to the Internet [5]. The problem of latency imbalance in the Internet will become more prominent for the end-to-end latency.

Delay-Based Geolocation. Delay-based geolocation is an important technique to build IP geolocation database used by online services to enforce geographic access policy and to determine advertising content based on clients' locations [15]. It relies on accurately measuring the minimum path latency to estimate the distance to the target. Under latency imbalance, since existing methods only consider latency inflation on single paths, not the difference between paths, the minimum path latency may be overestimated, degrading the accuracy of delay-based geolocation.

Clock Synchronization via NTP/SNTP. Network time protocol (NTP) and its counterpart, simple NTP (SNTP), are widely used for the clock synchronization of devices in the Internet [9, 16]. Since NTP and SNTP assume that paths are symmetric, its accuracy is greatly affected by path asymmetry. Latency imbalance, if measured separately on forward or return paths, is an indicator to path asymmetry. Moreover, latency imbalance implies that the accuracy of NTP/SNTP could be improved by using more symmetric paths.

Broader Impact. Besides delay-based geolocation, latency imbalance has an impact on all applications relying on measuring latency to estimate distance, e.g., network coordinate systems used in peer-to-peer systems [8, 17]. Moreover, latency imbalance would serve as an important metric to consider for the measurement studies examining latency characteristics.

2.2 Types of Load Balancing

Load balancing can be performed per flow, per destination, or per packet. In per-flow load balancing, load balancers commonly hash 5 fields in the packet header to determine the next hop to forward the packet, where the 5 fields are used as *flow identifier* and described as a five-tuple: ⟨source address, source port, destination address, destination port, protocol number⟩ [3, 18]. For the same source-destination pair, we can change port numbers to discover LB paths in between [4]. Per-destination load balancing allows routers to balance traffic based on the destination address. In per-packet load balancing, packets in the same flow could be forwarded to different paths causing reordering of packets. Besides the five-tuple in TCP and UDP packets, routers also load-balance ICMP packets based on their checksum fields [3]. Both per-flow and per-packet load balancing could cause latency imbalance between source-destination pairs. Per-flow load balancing affects about 40% of Internet paths, while per-packet load balancing only affects 2% [1]. We focus on per-flow load balancing in this paper, under which paths to the same destination can be controlled via port numbers.

2.3 Load-Balanced Segments or Diamonds

Starting from the source, paths to a given destination diverge at the first load balancer (*divergence point*) and eventually converge back (at the *convergence point*) to reach the destination. A multi-homed source could be a divergence point and similarly a multi-homed destination could be a convergence point. Following the literature, we refer to the multiple path segments between the divergence and convergence points (or *end points*) as the *load-balanced segments* or, in aggregate, as a *diamond*. Each diamond is defined by its divergence point (x_1) and convergence point (x_2) and referred to as the $[x_1, x_2]$ diamond. Since diamonds include the path segments where LB paths differ from each other, they can be used to dissect latency imbalance.

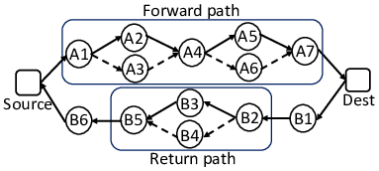


Fig. 1. An example of LB paths

Figure 1 shows an example of LB paths between a source and a destination. Due to path asymmetry, the forward and return paths are different and thus include different diamonds. In Figure 1, we have two diamonds, $[A1, A4]$ and $[A4, A7]$, on the forward paths and $[B2, B5]$ on the return paths. Suppose that the path segments $A1$-$A2$-$A4$ and $A1$-$A3$-$A4$ in $[A1, A4]$ differ in path latency. We refer to $[A1, A4]$ as *imbalanced diamond*, which are the causes for latency imbalance.

3 RELATED WORK

Since Augustin *et al.* presented the problems of *classic traceroute* under load balancing in 2006 [3], extensive efforts have been made to measure LB paths and study their impacts (e.g., [4, 11]). However, most of the efforts focus on the topology difference between LB paths, leaving the performance difference under-explored. Our objective is to characterize the latency difference between LB paths from a cloud-centric view and evaluate its impact on applications.

Discovering Load-Balanced Paths. *Paris Traceroute* [3] is the most popular tool to identify diamonds and enumerate LB paths between a source and a destination via the Multi-path Detection Alogrithm (MDA) [19]. To incentivize the deployment of Paris Traceroute and its MDA, Vermeulen *et al.* developed *MDA-Lite Paris Traceroute* incurring less network overhead [4]. Moreover, Almeida *et al.* extended MDA to measure load balancing in IPv6 networks [20] and Vanaubel *et al.* developed a method capable of discovering load balancing in MPLS tunnels [21]. Other than IP-level paths, load balancing was also studied at the AS level [1, 22]. Our work differs from these prior works in that we focus on the performance side of LB paths. Our work uses the topology of LB paths to discover diamonds as in [1, 4] and further extends the discovery of topologically invisible diamonds by observing their impact on latency imbalance between LB paths (§6.4.2).

Latency Difference Between LB Paths. In 2007, Augustin *et al.* conducted the first study on latency imbalance between LB paths for 22K source-destination pairs and only 2% were found to have significant latency imbalance [1]. To avoid the latency variability due to latency imbalance in latency measurement, *Tokyo ping* was developed as a replacement to `ping` [11]. Although both works are closely related to ours in that they found latency imbalance between LB paths and attempted to figure out the causes, our work differs in several aspects. First, we measure latency imbalance both in the cloud and from the cloud to the public Internet, while they have vantage points (VPs) and destinations all in the public Internet. Although virtual machines (VMs) were carefully excluded as sources in [11], we observe almost no impact of using VMs in the cloud as VPs on measurement accuracy (6.4). Second, their approaches to measure latency imbalance are either not scalable due to requiring path enumeration [1] or not systematic in choosing sample size [11]. We propose a lightweight approach to measure latency imbalance, which requires no path information and provides accuracy guarantees. Third, we evaluate the impact of latency imbalance on application performance and propose potential solutions. Our work also differs from [11] in that 1) we do not exclude last-hop routers from probing and observe no significant influence of slow-path ICMP generation on measured latency (§6.4), and that 2) we use UDP probes, though having less reachability than ICMP [23], to reflect more closely the latency imbalance experienced by the Internet traffic.

Applications. Latency imbalance implies that path selection may affect application performance. Wu *et al.* proposed to send multiple requests to explore the multi-path capability of load balancing to reduce latency inflation, where the first received request is considered taking the best path [24]. We evaluate how latency imbalance affects application performance on three applications: delay-based geolocation [7, 25], NTP [10] and VoIP [13]. Latency imbalance has impacts on latency-based client direction for global load balancing in content delivery networks [26, 27]. Latency imbalance could also affect many emerging applications, e.g., networked VR/AR [28], low-latency gaming [29] and industrial IoT [5]. Moreover, latency imbalance also makes us rethink the design of load balancing algorithms (e.g., hashing-based schemes [30]).

4 MEASURING LATENCY IMBALANCE

This section presents our focus and key challenges in measuring latency imbalance and our measurement methodology. We begin by defining latency imbalance.

Definition of Latency Imbalance. When LB paths exist between a source-destination pair, these paths could have different path latencies. To characterize the significance of the path latency difference, we define *latency imbalance* as the difference between the highest and lowest latencies of LB paths. We want latency imbalance to be stable and thus consider *path latency* as the minimum time needed for packets to travel through the path, excluding transient network dynamics (e.g., congestion). Such stable path latency is widely used in applications, e.g., Internet mapping [6], geolocation [7] and network clock synchronization [31].

4.1 Our Focus and Key Challenges

Our goal is to characterize latency imbalance at the Internet scale. We first discuss the exact type of latency imbalance to be measured, and then identify the key challenges and describe a strawman approach to highlight the limitations of existing measurement tools.

Measuring One- or Two-Way Latency Imbalance? Between a source-destination pair, latency imbalance could occur on both forward and return paths. *One-way imbalance* is the latency imbalance occurring only on forward paths, calculated as the difference between the highest and lowest one-way delays (OWDs) of LB forward paths. *Two-way imbalance* is the combination of latency imbalance on both forward and return paths. As we have no control over destinations, we can only use TTL-limited probes to obtain full topology on forward paths but not on return paths. While *Reverse Traceroute* can discover return paths using ICMP packets [32], the return paths cannot be pinned down by flow identifiers. This means that we can understand one-way imbalance with path analysis (§6.4 and §6.2), but cannot do the same for two-way imbalance. We thus focus on studying one-way latency imbalance in this paper, which affects all applications mentioned above in §2.1.

Key Challenges and A Strawman Approach. To measure one-way imbalance at the Internet scale, we face three challenges: 1) how to efficiently discover LB paths between each source-destination pair? 2) how to efficiently measure the stable latencies of LB paths for the calculation of latency imbalance? 3) how to measure one-way imbalance? A strawman approach is to first enumerate LB forward paths using *Paris Traceroute*, the most popular tool to enumerate LB paths to a given destination [3], and then measure their stable OWDs. However, the overhead of path enumeration is inherently heavy. Even with the lightweight version, MDA-Lite, recently developed by Vermeulen *et al.* [4], it could take several minutes to enumerate paths between a single source-destination pair for complex network topologies [33]. Moreover, OWD can be accurately measured

only if we have access to both ends of the path [34]. Access to a large number of hosts is needed for large-scale experiments. These issues make it difficult to scale the approach.

4.2 Overview of Our Methodology

As no existing measurement tool and public dataset are suitable for studying one-way imbalance, we develop our own measurement tool and use it to collect data by actively probing the Internet. Our tool is built on Yarrp [35] with functions (1000+ lines of code) added for the scheduling and processing of packets. To ensure global coverage, we send probes to IPv4 addresses from DCs around the globe. We initiate a prober process in each DC selected and the prober measures one-way imbalance to one representative address in each /24 prefix. Of 10.8M /24 prefixes probed, we were able to measure one-way imbalance to about 3M representative addresses (see Table 2). The prober has no access to the destination and thus measures one-way imbalance from a single end. The key ideas of achieving this include two points. First, we strategically manipulate the UDP probes to ensure the ICMP responses always traverse the same return path, thus eliminating the impact of return path load balancing. Second, we sample LB paths to the destination by sending UDP probes with different flow identifiers, specifically port numbers. Since the path of a UDP probe is passively selected by load balancers based on its flow identifier, for sufficiently large sample size, most of the LB paths would be covered with high probability (see §4.3.2). We use UDP rather than TCP probes, as we have no way of controlling the return paths of TCP responses (i.e., TCP ACKs or RSTs).

4.3 Measuring One-Way Imbalance

We want to efficiently measure one-way imbalance from a single end. This section first presents our methodology and then evaluates its accuracy in measuring one-way imbalance.

4.3.1 Fixing the Return Path. Each RTT sample measured at the prober is the sum of the traversal time of a UDP probe on the forward path and that of the corresponding ICMP response on the return path. For UDP probes sent to the same destination, we want to fix the return paths of ICMP responses such that the difference between RTTs is the same as the latency difference between forward paths. Since except for per-packet load balancers, ICMP responses are load-balanced based on their checksums [3], we fix the return path by enforcing ICMP responses to use the same checksum (see Appendix A). Per-packet load balancers on the return path could cause ICMP responses with the same checksum to take different paths, but only 2% of paths in the public Internet were found to traverse a per-packet load balancer in Augustin *et. al*'s 2007 survey [1]. In case there could be more per-packet load balancers in the public Internet nowadays or from the cloud to the public Internet, we further filter out unstable path latencies that could be affected by per-packet load balancing as in §4.3.4. Augustin *et al.* have proposed a general method of obtaining the desired ICMP checksum by manipulating the UDP data, which requires extra probing to the destination [1]. We simplify the method by sending UDP probes without UDP data, as detailed in Appendix A.

4.3.2 Achieving High Path Coverage with Probabilistic Guarantees. We refer to UDP samples with the same flow identifier as a *flow*. Instead of enumerating LB paths to a destination, we sample them by sending flows with different identifiers (specifically port numbers), which may take different paths to the destination. We want to determine the *sample size* (i.e., the number of flows) needed to achieve high path coverage.

Defining Path Coverage. Suppose there are n LB paths between a source-destination pair with latencies ordered as $l_1 \leq l_2 \leq \cdots \leq l_n$, and that the probability of taking the i-th path is p_i. We have m flow samples with the lowest and highest latencies denoted as l_Y and l_Z, where Y and Z are

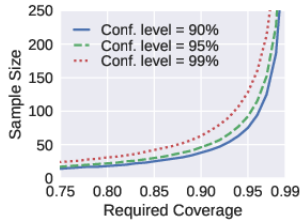


Fig. 2. Tradeoff between accuracy and efficiency

the path indices. Since latency imbalance is the difference between the lowest and highest latencies, all paths with latencies inclusively between l_Y and l_Z are covered. We thus define *path coverage* as the sum of the probability of the covered paths. Let $s[i, j]$ be the probability of taking a path with latency in interval $[l_i, l_j]$, i.e., $s[i, j] = \sum_i^j p_i$. The path coverage is then equal to $s[Y, Z]$. This definition of path coverage implicitly makes us favor covering the paths of high probability over those of low probability for high path coverage. As will be discussed later, a large number of extra samples is needed to cover a path of low probability with latency much lower or higher than other paths.

Probability of Achieving High Path Coverage. We want $s[Y, Z]$ to be higher than a threshold, called *required coverage*. We denote the required coverage as x and the probability of $s[Y, Z] > x$ as $p(s[Y, Z] > x)$. Let $w^+_{x,i}$ be the largest index with $s[i, w^+_{x,i}] \le x$ and $w^-_{x,i}$ be the lowest index with $s[w^-_{x,i}, i] \le x$. We can calculate $p(s[Y, Z] > x)$ as

$$\begin{aligned} & p(s[Y,Z] > x) = 1 - p(s[Y,Z] \le x) \\ & = 1 - p(s[Y,Z] \le x | Y \ge w^-_{x,n}) - p(s[Y,Z] \le x | Y < w^-_{x,n}) \\ & = 1 - s[w^-_{x,n}, n]^m - \sum_{i=1}^{w^-_{x,n}-1} p(Y=i) \left(\frac{s\left[i, w^+_{x,i}\right]}{s\left[i,n\right]} \right)^{m-1}, \end{aligned} \tag{1}$$

where $p(Y = i) = p(Y \ge i) - p(Y \ge i + 1) = s\left[i, n\right]^m - s\left[i + 1, n\right]^m$.

Coverage Guarantee and Tradeoff. We want to achieve high coverage with probabilistic guarantees. More formally, we want the actual coverage to be no less than the required coverage (x) with probability at least $1 - \delta$, i.e., $p(s[Y, Z] > x) \ge 1 - \delta$. Under the same δ, we need larger sample size to achieve higher path coverage; there is a tradeoff between the *measurement accuracy* (determined by path coverage) and the *measurement efficiency* (sample size). Given the required coverage and the probability distribution of paths (p_i's) between a source-destination pair, we can use Equation (1) to obtain the minimum sample size needed to achieve a certain probabilistic guarantee. In practice, however, as the probability distribution of paths is not known in advance, we cannot personalize the sample size for each pair. Instead, we choose a sample size that achieves the coverage guarantee for all source-destination pairs under various probability distributions of paths.

To collect a diverse range of LB paths, we measured LB paths between 232K source-destination pairs from 6 DCs belonging to 3 cloud providers to random IPv4 addresses. For each pair, we approximate the probability distribution of paths using their frequency distribution [36]. More specifically, we measure the IP-level paths of 600 flows with different identifiers for each pair and count the frequency of each path. The normalized frequency of a path is used as its probability. Figure 2 shows the sample sizes needed to achieve different required coverages with confidence levels of 90%, 95% and 99% respectively. Although the sample size increases exponentially with the

required coverage, the increase rate is low when the required coverage is under 0.85 and we can use 30 samples to achieve the required coverage of 85% at the 95% confidence level.

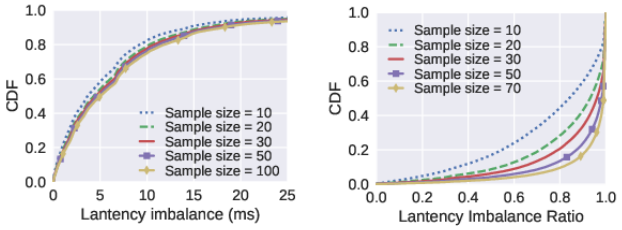
It is worth noting that cloud providers may place many load balancers close to DCs to distribute high-volume traffic and these load balancers provide a large number of unique interfaces, resulting in hundreds of IP-level LB paths between DCs and destinations. The probability distributions of LB paths from Amazon's and Alibaba's DCs to destinations are nearly uniform. Only < 1% of the DC-destination pairs fail the chi-squared test for uniformity at the 95% significance level. The large number of LB paths further necessitates the use of sampling to measure latency imbalance from a cloud-centric view.

4.3.3 Obtaining Stable Latencies for Latency Imbalance. We refer to flows with the highest and lowest latencies as *extreme flows*. Latency imbalance is then the latency difference between extreme flows. A simple approach to calculating latency imbalance is to first measure stable latency to each flow and then select the extreme flows. To reduce measurement overhead, we only measure flows that could potentially be the extreme flows. We first sample each flow once and select the flow with the highest RTT as the candidate for the highest-latency flow. Since the RTT of the candidate could possibly be inflated, we probe the candidate for six consecutive rounds and consider the candidate is the highest-latency flow if its RTT maintains the minimum over the six rounds. We show in Appendix E that using six probes can effectively mitigate the effects of latency inflation. If the flow's RTT is higher than that in any of the six rounds, we update the flow's RTT to the minimum RTT in the six rounds to mitigate inflation. We then choose the next flow with the highest RTT among all flows and probe it over another six rounds until the highest-latency flow is found. Similarly, we can determine the lowest-latency flow. It is possible that the RTTs to some addresses are highly varying and the extreme flows cannot be determined within a reasonable amount of probes. We stop probing an address if no extreme flows could be determined after 90 rounds. After probing, each extreme flow will have at least 7 RTTs, which will be used to prune probing results.

4.3.4 Pruning Probing Results. Although the minimum RTT of a flow is conventionally used as its latency in existing methods [6], it cannot eliminate latency inflation. Moreover, it could cause flow latency to be underestimated due to measurement errors, i.e., the minimum RTT is an outlier much lower than other RTTs of the flow [6]. The inflation of the highest flow latency and the underestimation of the lowest flow latency both result in the overestimation of latency imbalance. We further prune probing results as below. Flows with inflated latencies typically have varying RTTs over time. We use the variability of RTTs to prune the highest-latency flows with unstable latencies. The variability of RTTs is measured by the median absolute difference (MAD), robust to outliers [37]. We empirically prune a highest-latency flow if it has a MAD > 2% of the flow's latency. For the lowest-latency flow, we detect measurement errors by checking if the flow's minimum RTT is much lower than other RTTs. We pruned about 5% of lowest-latency flows with the minimum RTT lower than the second minimum one for > 5% of the flow latency.

4.3.5 Impact of Sampling on Accuracy and Sample Size Selection. To understand how sampling affects accuracy, we want to know the distance between the maximum latency imbalance and the measured one. We begin by estimating the maximum latency imbalance.

Estimating the Maximum Latency Imbalance. Obtaining the maximum latency imbalance requires enumerating all LB paths and measuring their path latencies. When 100s of LB paths are common between source-destination pairs, it is impractical to directly measure the maximum imbalance for a large number of destinations. Instead, we approximate the maximum latency imbalance using our method with a large sample size. When the sample size is 100, we can achieve path coverage greater than 95% at the 97% confidence level. In other words, the probability of



(a) Aggregate Imbalance (b) Pairwise imbalance

Fig. 3. Impact of sampling on accuracy

the uncovered paths only sums up to $\leq 5\%$ at the 97% confidence level. We thus consider these uncovered paths to be rare. We use the latency imbalance measured with 100 flow samples to approximate the maximum latency imbalance and study the accuracy loss due to sampling.

Accuracy of Sampling and Sample Size Selection. We evaluate the accuracy of sampling in measuring two types of latency imbalance: 1) *aggregate imbalance*, the collection of latency imbalance for a large number of source-destination pairs and 2) *pairwise imbalance*, latency imbalance for single source-destination pairs. In our experiments, we measure latency imbalance with different sample sizes back to back for the same source-destination pair, where a pair is discarded if path changes are detected while the path is being measured. We use CDF to characterize the distribution of latency imbalance in aggregate and measure the *difference between CDFs* as $|F_1(x) - F_2(x)|$, where $F_1(x)$ and $F_2(x)$ are two CDFs of latency imbalance.

Figure 3(a) shows the CDFs of latency imbalance under different sample sizes for 469K source-destination pairs from 6 DCs to random IPv4 addresses. Sampling is in general effective in measuring the aggregate imbalance. Even with 10 samples, the measured CDF only has at most 11% difference from that measured with 100 samples. The maximum difference decreases to 4% for 30 samples and further decreases to 2% for 50 samples. Moreover, the difference between CDFs reaches the maximum at low latency imbalance (< 3ms) and decreases as the latency imbalance increases. Considering both accuracy and efficiency, we use 30 samples when measuring aggregate latency imbalance.

For each source-destination pair, we calculate the ratio of the latency imbalance measured with each sample size to that with 100 samples respectively. Figure 3(b) shows the distribution of the ratios under different sample sizes. For 10 samples, only 40% of pairs have a ratio > 0.9. The percentage increases with the sample size and reaches 80% for 70 samples. Compared to aggregate imbalance, we apparently need a much larger sample size to accurately measure pairwise imbalance. We use 100 samples to measure inter-DC latency imbalance in §7.

4.4 Source and Destination Selection

Source Selection. We probe from 16 DCs in 14 cities around the globe, including 4 in Europe, 3 in North America, 6 in Asia and 1 in Australia. All but four of the DCs are located in different cities. Of the four, two are located in London and belong to different cloud providers, and the other two are located in Sydney, also belonging to different cloud providers. We intend to use these four DCs to observe the effect of cloud provider choices (§6.2).

Destination Selection. Considering the similarity of addresses in the same /24 prefix [38], we probe only one responsive address for each /24 prefix to reduce measurement overhead. For /24 prefixes without any responding addresses, we use the responding router closest to the destination as a *proxy* for the entire /24. To ensure that probes to the proxy follow the same path as probes to

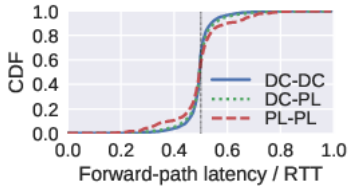


Fig. 4. Distribution of path asymmetry

the intended /24 prefix, the probes are directed to a (non-responding) destination within the /24 prefix with TTLs expiring at the proxy.

4.5 Metrics for One-Way Imbalance

Absolute and Relative Latency Imbalance. The latency imbalance defined above is the absolute latency difference between the highest- and lowest-latency paths, called *absolute imbalance*. However, the same absolute imbalance is of higher significance for low path latency than for high path latency. To capture the relative significance of absolute imbalance to path latency, we introduce a new metric called *relative imbalance*, which is the ratio of absolute imbalance to the forward-path latency of the highest-latency path. Relative imbalance is calculated with respect to the forward-path latency because one-way imbalance only occurs on forward paths. Relative imbalance can be *interpreted* as the percentage of latency reduction on the forward paths by switching from the highest-latency flow to the lowest-latency one. As forward-path latency is OWD, which cannot be accurately measured from a single end, we thus estimate forward-path latency from its RTT.

Estimating Forward-Path Latency. In 2008, Pathak *et al.* showed that estimating OWD of a path as 0.6 times its RTT would overestimate the OWD in 90% of cases, due to path asymmetry [34]. Conservatively, we want to use 0.6 times RTT of the highest-latency flow as its forward-path latency in computing relative imbalance to avoid overestimation. To verify that their results still apply in the current Internet, especially when data center networks are involved, we re-did the experiments in 2019 using public instances in 24 DCs and 31 PlanetLab (PL) nodes that are geographically distributed. An NTP daemon runs on each node to synchronize its local clock and estimates the synchronization error. A sufficient number of flows are exchanged between each pair of nodes to explore alternate paths in between. As in [34], flows with a synchronization error > 3% of their RTTs are pruned. Figure 4 shows the distribution of path asymmetry for 35,328 flows from 552 DC-DC pairs, 85,910 flows from 1,354 DC-PL pairs and 48,747 flows from 765 PL-PL pairs. PL-PL pairs generally have the worst path asymmetry, but still have 90% of flows with the ratio of forward-path latency to its RTT > 0.6. This confirms that using 0.6 times RTT of a path to conservatively estimate its OWD is still reasonable in the current Internet. We use "absolute imbalance" and "relative imbalance" to mean absolute and relative one-way imbalance by default unless stated otherwise.

5 IMPACT OF ONE-WAY IMBALANCE ON APPLICATIONS AND BASELINE

Before looking at the measurement results on one-way imbalance, we first want to intuitively understand its impact on applications. We take three applications (in Table 1) as examples and discuss how latency imbalance affects their performance. We set a *baseline* of one-way imbalance for each application to help evaluate the significance of the impact.

Clock Synchronization by NTP. To synchronize time, an NTP client estimates the NTP server's clock and adjusts its local clock to the estimated server's clock. The NTP client sends a message to the server and estimates the server's clock as the sending time of the response plus the estimated OWD from the server to the client. Since the OWD is typically estimated to be half

Table 1. Applications Affected by Imbalance and Baseline

Applications	Latency (OWD)	Accuracy	One-Way Imbalance Baseline
NTP	10s of ms	1s of ms	max(1ms, 5% of OWD)
Geolocation	10s of ms	< 400km	2ms
VoIP (Intl.)	100s of ms	×	20% of OWD

of the RTT, a major source of error for NTP is path asymmetry [39]. We define $\overline{\Delta}_d$ as the average of one-way imbalance on the forward and return paths. Through both analysis and experiments (see Appendix B), we show that the maximum reduction in synchronization error by considering latency imbalance is $\leq \overline{\Delta}_d$ for all source-destination pairs and $> \overline{\Delta}_d/2$ for most of the pairs.

A study on NTP servers shows that the OWDs between most NTP servers and their peers (other servers used for synchronization) are in the 10s of milliseconds and that the synchronization errors are in the 1s of milliseconds [10]. We simply use 10% of OWDs as reference for typical synchronization errors, where the error is proportional to OWD because longer distance between NTP servers and clients is likely to result in larger synchronization error. We consider *significant* error reduction to be > 25% of the synchronization error, i.e., 2.5% of OWD. Since the maximum error reduction is mostly at least half the amount of one-way imbalance, we set the baseline of one-way imbalance to the maximum of 1ms and 5% of OWD, where 1ms is used when OWDs are low.

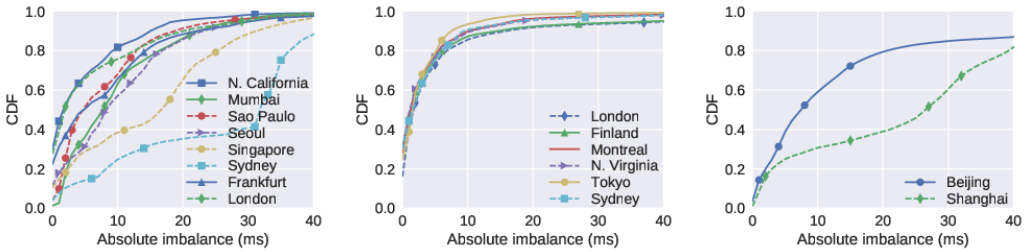
Delay-Based Geolocation. Delay-based geolocation is an important technique to build IP geolocation database used by online services to enforce access policy and to determine advertising content [15]. To geolocate an Internet host, delay-based geolocation uses a set of hosts with known locations, called *landmarks*, and measure the latency between each landmark and the host. The measured latency is then used to estimate the distance (r) to the host. From the perspective of each landmark, the host is in a circle with radius r centered at the landmark. The intersection of these circles is the estimated geolocation region of the host. We can see from this process that larger-than-actual latency causes overestimated distance, resulting in a larger estimated geolocation region. Since city-level accuracy is an important metric to the existing geolocation databases [15], we want to the overestimated distance to be at least < 400km, where 400km is simply used to approximate the diameter of the largest city in the world. We use the worst case to set the baseline such that latency imbalance higher than the baseline would be *significant* to delay-based geolocation. Using the travel speed of signals in fiber-optic cable (2/3 of the speed of light in vacuum), we obtain that it takes 2ms for signals to travel 400km. We thus set the baseline to be 2ms. Since landmarks in the same subcontinental region (e.g., country) of the target host are preferred for better accuracy, we use the average OWD in North America (in the 10s of milliseconds) as a reference for the typical latency in delay-based geolocation [40].

VoIP. International VoIP calls have been rising in recent years and have taken a large portion of Internet-based calls (about 46% of Skype calls are international [13]). However, international calls are more likely to experience high RTT and have poor call quality. According to [13], about 5% of Skype calls use paths with RTT over 400ms. For these calls, the fraction of calls with poor quality (with a rating of 1 or 2) can be reduced by > 10% when OWD is decreased by 20%. We set the baseline to be 20% of OWD and consider one-way imbalance larger than the baseline to be *significant* to call quality. Since call quality is more sensitive to latency reduction at high path latencies than at low ones, we would expect that latency imbalance have more significant impacts on calls experiencing high latency. We use 100s of milliseconds as the latency of interest for VoIP.

Table 2. /24 Address Prefix Reachability

Responsive /24s			Proxies			Probed /24s
Stable	Pruned	Expired*	Stable	Pruned	Expired	
2.7M	598K	736K	254K	92K	29K	10.8M

* No extreme flows can be obtained within 90 probes.



(a) Amazon EC2 (b) Google cloud (c) Alibaba cloud

Fig. 5. Distribution of relative imbalance from DCs to public IPv4 addresses.

Other Applications. Other applications vulnerable to high imbalance include networked AR/VR [28], low-latency gaming [29], real-time video conferencing [41], industrial IoT [5], etc.

Baseline. The baseline for each application is summarized in Table 1. Latency imbalance affects applications with various operating latencies and applications have different degrees of sensitivity to latency imbalance.

6 IMBALANCE FROM DATA CENTERS TO PUBLIC IPV4 ADDRESSES

We probed from the 16 DCs to 10.8M /24 prefixes covered in the CAIDA's *IPv4 prefix-to-AS mapping* database [42]. The instance types we used in Amazon's, Google's and Alibaba's DCs were c5.large, n1-standard-2 and ecs.ic5.large respectively, all of which had 2 virtual CPUs and at least 2GB memory. Table 2 characterizes these /24 prefixes by the degree to which we were able to reach them. A /24 prefix is considered "Responsive" if at least one address within the address space responded to our probe. We are unable to determine imbalance for 736K /24 prefixes and 29K proxies (termed "Expired") because we cannot determine the extreme flows after sending 90 probes. About 598K prefixes and 92K proxies are pruned because their extreme flows (defined in §4.3) have unstable latencies. In total, we have latency imbalance measured to roughly 2.7M /24 prefixes and 254K proxies. The rest 6.4M /24 prefixes are not measurable in that we were not able to reach a responsive host and we could not establish 30 flows to the last-responding router proxy.

In the following, we report only on the results of probing the 2.7M /24 prefixes and 254K proxies with stable imbalance from the 16 DCs, or about 48M source-destination pairs in total. These /24 prefixes are widely spread covering 96% of countries in the world and 68% of ASs in the CAIDA's prefix-to-AS database.

6.1 Distribution of One-Way Imbalance

Absolute Imbalance. Figure 5 shows the distribution of absolute imbalance seen from the perspective of each DC. Latency imbalance is prevalent between DCs and public addresses. Even the DC with the narrowest distribution (Google Tokyo) has absolute imbalance > 5ms to about 20% of probed addresses. Moreover, latency imbalance has a wide range; the distributions of absolute imbalance for most DCs are long-tailed. The narrowest distribution (Google Tokyo) has a tail

Fig. 6. Distribution of relative imbalance under different ranges of OWDs

covering 7% of addresses with absolute imbalance > 10ms. The absolute imbalance for Amazon's Sydney and Alibaba's Shanghai DCs are more widely distributed with shorter but heavier tails and have absolute imbalance > 20ms to more than 60% of addresses. Moreover, latency imbalance differs significantly between cloud providers, as will be studied further in §6.2.

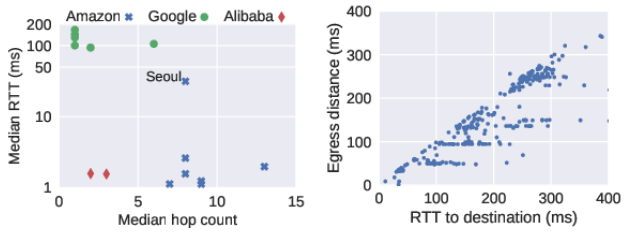
Relative Imbalance. Figure 6 shows the distribution of relative imbalance under different ranges of OWDs for 5 DCs. We do not compute relative imbalance for low OWDs (< 20ms) to mitigate the effects of measurement errors and our dataset only includes 5% of source-destination pairs with OWD < 20ms. Figure 6 shows that Amazon's Sydney and Alibaba's Shanghai DCs, with the highest two absolute imbalance, have most of the high imbalance occurring at high path latencies (OWD >120ms). If we only consider addresses with OWD > 120ms from DCs, Amazon's Sydney and Alibaba's Shanghai DCs have relative imbalance > 0.2 to about 26% and 41% of them, and the other 3 DCs also have relative imbalance > 0.2 to 7%, 13% and 21% of them respectively. This implies that traffic in the WANs between DCs and public addresses would experience significantly less latency by choosing a path of lower latency. Potential applications include using DCs as relays for international VoIP calls [13] or as globally accessible cloud VPN [43].

Besides imbalance in high path latencies, most DCs also have large relative imbalance to a significant portion of addresses with low path latencies (OWD < 60ms). Except for Amazon's Sydney DC, other DCs have relative imbalance > 0.1 to about 6% (Amazon London) to 13% (Alibaba Beijing) of all reported addresses. Amazon's Sydney DC has nearly zero relative imbalance in low path latencies due to a small number of responsive domestic addresses (< 10K). If we only consider addresses with OWD < 60ms from DCs, Amazon's and Google's London DCs have relative imbalance > 0.1 to about one-fifth of them, and Alibaba's Beijing and Shanghai DCs have imbalance > 0.1 to no less than one-third of them. This implies that the accuracy of delay-based geolocation and NTP could both be improved for a large portion of addresses by using lower-latency paths. Potential applications include geolocating cloud VPN servers [7] and time synchronization for cloud-enabled IoT, e.g., real-time earthquake detection [44].

6.2 Why Cloud Providers Differ in Latency Imbalance?

As shown in Figure 5, Google and Amazon both have DCs in Sydney and London, but Google's Sydney DC has much lower imbalance than Amazon's Sydney DC. To understand the causes, we divide the distance from a DC to a destination into 1) the distance from the DC to the first router in its neighbor AS, called *egress distance*, and 2) that from the router to the destination. Since cloud providers operate DCs and routers within the egress distance in their own ASes, a longer egress distance indicates more control over the entire path to the destination. We will show that Google's DCs have much longer egress distances than other cloud providers' DCs.

Given the path from a DC to a destination, the difficulty in determining the egress distance is to locate which router along the path is the border router of the DC's neighbor AS. Inferring the border routers of ASes is a complicated problem and simply mapping a router to the origin



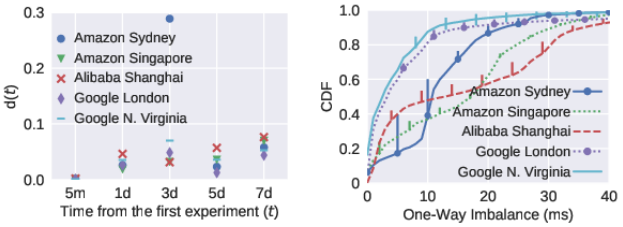
(a) Distance to neighboring ASs (b) Egress distance of Google Tokyo

Fig. 7. Imbalance difference between cloud providers

AS of its interfaces could be incorrect for many reasons [45]. To obtain accurate router-to-AS mapping, we considered using CAIDA's ITDK datasets including Internet-scale routers' ownership, inferred from traceroute paths from vantage points to destinations. However, as traceroute discovers router interfaces facing the vantage points, only about 20% of router interfaces in our dataset are covered by the ITDK dataset [46], where our dataset includes the traceroute paths from each DC to responsive addresses. We thus use *bdrmapIT* to obtain routers' ownership using our dataset, together with other inputs including prefix-to-AS mapping supplemented with RIR delegations [47–51], IXP prefixes [52–54] and AS relationships [55]. Using the derived router-to-AS mapping, we first identify the interdomain link (the near side in the DC's AS and the far side in its neighbor AS) from the traceroute path for each DC-destination pair and then measure the egress distance in both hop count and RTT.

Figure 7(a) shows the median egress distance (measured in hop count and RTT) from each DC to responsive addresses. All Google DCs have the median RTT > 94ms, while Amazon's and Alibaba's DCs (except for Amazon Seoul) have the median RTT < 3ms. Looking into the egress distances of Google's DCs, we found that egress distance generally increases with the distance (measured in RTT) from the DC to destinations and is capped at certain values for some destinations. Figure 7(b) shows the relation between egress distance and distance to destinations for 1,000 destinations seen from Google Tokyo. The egress distance is capped at around 50ms, 100ms and 140ms for some destinations. This indicates that Google uses its own private WANs to route packets to an egress point close to traffic destinations. This observation is further confirmed by Google's own documentation on network service tiers [56], where traffic in premium network tiers is forwarded as close to the destination's ISP and all our VMs in Google cloud are in premium network tiers. As a result, the imbalance incurred in Google's own networks dominates the latency imbalance for the entire paths. For instance, for 77% of destinations probed from Google Tokyo, more than 60% of the entire path latency is spent on travelling from the DC to its neighbor ASes. Despite the long egress distance, the latency imbalance incurred by Google's networks is low. The worst DC has absolute imbalance < 10ms before entering its neighbor ASes for at least 88% of destinations. This shows that the low imbalance of Google's DCs is due to the use of well-balanced long-distance paths in Google's own networks.

As shown in Figure 7(a), Amazon's and Alibaba's DCs are close to neighbor ASes and thus should forward packets to the public Internet close to traffic sources. The low egress distance of Amazon's and Alibaba's DCs results in low latency imbalance to neighbor ASes, where all of these DCs have the median absolute imbalance < 0.8ms. Amazon's Seoul DC has the median RTT equal to 32ms, but the median absolute imbalance to neighboring ASs is only 0.2ms. Since the latency imbalance before entering the neighbor ASs is low, we can conclude that the major imbalance of Amazon's and Alibaba's DCs to destinations occurs in the public Internet.



(a) Latency imbalance over time (b) Distribution of latency imbalance

Fig. 8. Stability of latency imbalance over time

6.3 Is Latency Imbalance Stable Over Time?

We consider *aggregate latency imbalance*, which is a collection of latency imbalance measured to a set of destinations in a single experiment. We want to know if aggregate latency imbalance is stable over experiments. Specifically, for each DC, we first measure latency imbalance to the same set of destinations in experiments starting at different times. We then use the *first* experiment as reference and compare the aggregate latency imbalance in subsequent experiments respectively with that in the first experiment to check if the difference increases with time or remains stable. We use a CDF to characterize the distribution of latency imbalance in aggregate and measure the *difference between aggregate latency imbalances* by the maximum difference between their CDFs. Suppose the first experiment starts at time 0 and that $F_t(x)$ denotes the percentage of destinations with latency imbalance $\leq x$ in the experiment starting at time t. The difference between CDFs of latency imbalance in the first experiment and the experiment at time t is then $d(x,t) = F_t(x) - F_0(x)$ with the maximum equal to $d_{max}(t) = \max_{x \geq 0} |d(x,t)|$ over all latency imbalance.

Figure 8(a) shows the $d_{max}(t)$'s for 5 representative DCs in experiments starting at different t's over a week, where latency imbalance is measured from each DC to 200K[1] destinations. Except for Amazon Sydney, all other DCs have stable distributions of latency imbalance across experiments with $d_{max}(t) < 0.08$ for all t's. We further examine if $d_{max}(t)$'s are reached at low or high latency imbalance. Figure 8(b) shows the distribution of latency imbalance in the first experiment of the week with error bars being $d_{max}(x) = \max_{t>0} |d(x,t)|$, the maximum difference between CDFs of latency imbalance over all experiments in the week at latency imbalance equal to x. We find that the differences between experiments are large (the high error bars) mostly at low latency imbalance and decrease as latency imbalance increases. From Figure 8(b), we thus can conclude that except for Amazon Sydney, other DCs have stable latency imbalance for at least one week. To check if these DCs have stable latency imbalance for longer periods of time, we compare the first experiment in Figure 8 (starting on September 1, 2019) with that in Figure 5, conducted in February, 2019. We find that these DCs (except for Amazon Sydney) still have small $d_{max}(t) < 0.10$ occurring at low latency imbalance. This indicates that most of the DCs have stable aggregate latency imbalance even over months.

Compared to other DCs, Amazon Sydney has more variable aggregate latency imbalance due to changing network conditions. At t = Day 1, $d_{max}(t)$ is reached at latency imbalance equal to 9ms. Comparing the destinations with latency imbalance around 9ms (specifically from 8 to 11ms) in all experiments for Amazon Sydney, we found that the experiment at Day 1 only shared < 20% of these destinations with other experiments, while other experiments shared > 80% of these

[1]We conduct our medium-sized experiments at the scale of 100s of thousands of destinations, a typical scale for many existing measurement campaigns [4, 57]. As a reference, 200K random destinations cover roughly about 87% of countries, 13K ASes.

destinations among themselves. Considering all experiments probe to the same set of destinations, this indicates that the network conditions from Amazon Sydney to a large number of destinations changed temporarily at t = Day 1 and reverted back to the previous conditions at later days. The large difference between Amazon Sydney's latency imbalance in Figure 8 and Figure 5 is likely due to network condition changes (e.g., caused by traffic engineering) or the use of different source addresses in the two experiments, resulting in LB paths traversing different parts of the Internet.

6.4 Diving Deeper Into One-Way Imbalance

As mentioned in §2.3, imbalanced diamonds with unequal latencies on their path segments are the causes for one-way imbalance. In this section, we want to 1) verify that our measured one-way imbalance is indeed due to imbalanced diamonds, 2) analyze imbalanced diamonds to understand the causes, and 3) discuss the challenges of load balancing in terms of latency imbalance.

6.4.1 Verifying Our Measured One-Way Imbalance. We want to verify that our measured one-way imbalance reflects the imbalance experienced by LB paths rather than measurement artifacts. We first collected 10K samples of one-way imbalance from each prober to its nearby routers at different times throughout our experiments (in §6.1) and found that the average one-way imbalance is < 0.2ms for all probers. This confirms that the probers have almost no impact on the measured imbalance. Measurement errors could also come from the destinations due to 1) slow ICMP responses, generated in the slow-path [11], and 2) irregular router behaviors, e.g., adding IP options to ICMP responses or modifying their contents [1]. It is challenging to directly estimate these errors [58] and we instead verify the accuracy of our measured imbalance to a destination using its last-hop router(s) as follows.

The key idea is that if latency imbalance between the extreme flows (i.e., the highest- and lowest-latency flows) is caused by traversing an imbalanced diamond, the latency imbalance will be carried to hops after the diamond. In other words, if our measured imbalance is due to imbalanced diamonds, we would expect the latency differences between the extreme flows seen at a destination (denoted as I_1) and its last-hop router(s) (denoted as I_2) to be similar, unless non-negligible latency imbalance occurs in between. In our experiments, we only use destinations in the same /24 subnet as their last-hop routers, where the distance between them is likely to be close and results in negligible imbalance. We define the *error* of the measured imbalance as the absolute difference between I_1 and I_2. We found that our measured imbalance had an error < 1ms for 95% of source-destination pairs of all DCs, where the worst DC (Google Tokyo) had an error < 1ms for 81% of pairs and an error < 2ms for 89%. Looking into the worst DC, we confirmed that accuracy of path latency was not an issue for the pairs with an error > 1ms, because the back-to-back imbalance samples to the same destination had a difference < 0.5ms for all destinations. We further checked that only 1% of these pairs experienced ICMP responses with different checksums. The error thus should be mainly due to the imbalance between the destinations and their last-hop routers. This implies that our measured imbalance should have higher accuracy than the reported. Since from the perspective of destinations, the imbalance possibly caused by middleboxes or firewalls is the same as that caused by unequal path lengths, we consider them as part of imbalanced diamonds.

6.4.2 Top Imbalanced Diamonds and Potential Causes. We first discuss how to discover imbalanced diamonds and then the potential causes for imbalanced diamonds.

Discovering Imbalanced Diamonds. Figure 9 shows an example of discovering imbalanced diamonds. By merging the common hops of the extreme paths (solid and dashed lines) in the forward direction, we can identify diamond $[A1, A4]$ from the topology. Suppose we want to estimate the *imbalance of a diamond*, i.e., the latency difference between the diamond's path segments. We first

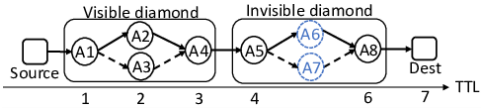
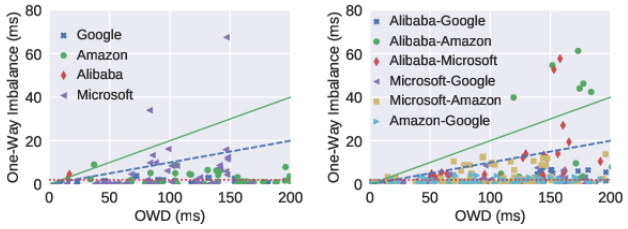


Fig. 9. Visible and invisible diamonds

measure the latency difference between the extreme paths at both the convergence and divergence points of the diamond, denoted as d_1 and d_2 respectively. The difference between d_1 and d_2 is then the change in latency difference due to traversing the diamond, equal to the imbalance of the diamond. However, not all diamonds can be discovered by topology. Routers could be unresponsive or even invisible to TTL-limited probes, e.g., routers in MPLS networks [59]. We refer to diamonds discoverable by topology as *visible diamonds* and those not as *invisible diamonds*. Diamond [$A5, A8$] in Figure 9 is an invisible diamond with unresponsive routers $A6$ and $A7$. We identify an invisible imbalanced diamond from its imbalance.

Imbalanced Diamonds and Potential Causes. For each source-destination pair, we first discover visible diamonds and group the hops in the rest of path segments into pairs of two, e.g., $A4$-$A5$-$A8$ in Figure 9 is divided into $A4$-$A5$ and $A5$-$A8$. We consider each pair as a potential invisible diamond and calculate its imbalance. Considering that the error of our measured imbalance is $<$ 1ms at one hop (§6.4.1), we would expect an error at most 2ms for diamond imbalance. We use a relative large threshold, 5ms, to determine imbalanced diamonds and found 4.1M visible and 2.1M invisible imbalanced diamonds. Although invisible diamonds can be identified from the topology, it is error-prone to identify the actual end points of a diamond at the IP level due to IP aliasing [1]. Since we do not use alias resolution techniques, we limit our focus on invisible diamonds with their end points having consecutive TTLs such that no aliases exist in the diamonds. We further require each invisible diamond to have at least 10 samples and use the median sample as the diamond imbalance. After pruning, we obtain 65K invisible imbalanced diamonds, where 32% have imbalance $<$ 10ms and 50% have imbalance between 10ms and 20ms. For such invisible diamonds, load balancing between their end points could occur between multiple paths in MPLS networks [21] and between links in link aggregation groups (LAGs) [11].

Using the database of router ownership in §6.2, we can categorize these diamonds into two types: 1) *inter-AS*, if the convergence and divergence points are in different ASes and 2) *intra-AS*, otherwise. We found that 80% of these diamonds are intra-AS and that almost half of these intra-AS diamonds are from the top 10 imbalanced ASes (containing the most number of intra-AS diamonds), including 4 Tier-1 providers: AS174 (Cogent), AS7018 (ATT-INT), AS2914 (NTT America) and AS6762 (Telecom Italia). We confirmed that 7 of the top 10 imbalanced ASes use MPLS from existing studies of MPLS networks [59, 60]. Further, invisible diamonds directly between border routers were found in all of these ASes except AS7922 (Comcast Backbone) and AS4230 (Embratel). Since MPLS tunnels are widely used inside ASes to connect border routers [21], these invisible diamonds are very likely due to invisible MPLS tunnels with imbalanced load balancing. AS12389 (Rostelecom) and AS6762 have respectively 68% and 100% of invisible diamonds between border routers, which could indicate wide deployment of invisible MPLS tunnels. Other than invisible MPLS tunnels directly between border routers, we can also see invisible MPLS tunnels with the last hop of the egress border router visible to traceroute when the Penultimate Hop Poping (PHP) is activated [59]. We found that 6% and 30% of invisible diamonds in AS4230 and AS12389 respectively used such invisible MPLS tunnels. Identifying other types of diamonds requires careful probing analysis [61] and validation requires information from the operators. Both are subject to future research. We also found 8,601 inter-AS invisible diamonds spreading across 3,340 pairs of ASes, of which 7.6% are peers. This indicates that latency imbalance also occurs between border routers of inter-domain



(a) Intra-cloud (b) Inter-cloud

Fig. 10. Intra-cloud and inter-cloud latency imbalance

links. Both LAG and multi-paths in invisible MPLS networks are the possible causes for the latency imbalance.

Challenges of Load Balancing. The definitions of path cost are diverse and a certain definition of path cost (e.g., latency) can at best ensure the related performance metric to be equal across LB paths. However, applications have a wide range of requirements from latency- to bandwidth-sensitive. In addition to changing network conditions, it is impossible to have LB paths with equal performance for all applications. LB paths optimized for equal bandwidth may differ greatly in latency [62]. When performance imbalance is significant, undifferentiated selection of LB paths, as done by hashing-based algorithms, would degrade application QoE. Latency imbalance emphasizes the importance of differentiated services. Diffserv-aware MPLS traffic engineering (MPLS-TE) is a good option for providing differentiated services in the Internet. However, MPLS-TE generally monitors and computes label switching paths at the router level, while load balancing could occur at the link level [21]. Since multiple links could exist between routers, it is challenging to monitor link-level path performance.

7 IMBALANCE BETWEEN DATA CENTERS

Inter-DC WANs, e.g., Microsoft's SWAN [63] and Google's B4 [64], are designed to carry traffic between DCs for various interactive services highly sensitive to latency [63]. We thus want to understand the latency imbalance between DCs.

7.1 Data Collection

To have a more rounded view of latency imbalance between DCs, we added 8 Microsoft Azure's DCs located in New South Wales (Australia East), Tokyo, Chennai (South India), California, Virginia, Toronto, London, and Paris. We refer to two connected DCs as a *DC pair*. Due to path asymmetry, connectivity of DC pairs is *directional*. We denote the set of paths from DC a to DC b as $a \mapsto b$. Along with the 16 DCs in Figure 5, we have 552 DC pairs amongst the 24 DCs. We sample the latency imbalance between each pair every 20 minutes for one day using the same methodology in §4. We use 100 flows for each source-destination pair as studied in §7. No latency imbalance sample can be obtained if no extreme flows are found within the first 160 probes or extreme flows have unstable latency. After the data pruning in §4.3, we have 5 pairs with < 10 samples left. We prune these pairs and analyze the stability of latency imbalance for the rest of pairs. To avoid outliers, we use the median latency imbalance sample as representative for each DC pair in our study.

7.2 Intra- and Inter-Cloud Latency Imbalance

We separate DC pairs into intra- and inter-cloud pairs for analysis. An intra-cloud pair is between DCs belonging to the same cloud provider, while an inter-cloud pair consists of two DCs from different cloud providers.

7.2.1 Intra-Cloud Latency Imbalance. Figure 10(a) shows the one-way imbalance of all intra-cloud pairs with respect to their OWDs, where the three lines present one-way imbalance equal to 2ms, 10% of OWD and 20% of OWD respectively. In general, Google has the least latency imbalance with only 10% of its intra-cloud pairs having imbalance slightly > 2ms, while Microsoft has 7 out of 56 pairs with imbalance > 10% of their OWDs. Of the 7 pairs, 2 pairs have imbalance as high as 34ms and 67ms, much larger than 20% of their respective OWDs, and they have very stable latency imbalance over time. The high latency imbalance would cause Microsoft's traffic between the same DC pair to experience disparate path latencies, causing inconsistent performance for clients in latency-sensitive applications. We only have two intra-cloud pairs between Beijing and Shanghai for Alibaba with OWD around 15ms and one-way imbalance > 2ms.

We further analyze the paths between DCs as in §6.2. We found that all intra-cloud pairs have all discovered intermediate hops in the ASs of their respective cloud providers. As Google's private WAN is well-balanced, its intra-cloud pairs have low latency imbalance. Looking into Microsoft's pairs with high imbalance (> 10% of OWD), we found that 1) two pairs with the highest imbalance use flows of disparate path lengths (measured in the total hop count), where the lowest-latency flow takes a path of length only about half of that of the highest-latency flow and that 2) even when paths have similar lengths, the long-distance inter-continental paths are not well-balanced and cause most of the imbalance.

7.2.2 Inter-Cloud Latency Imbalance. The percentage of inter-cloud pairs with high imbalance (> 10% of OWD) is similar to that of intra-cloud pairs. Pairs involving Google's DCs generally have lower latency imbalance than other pairs, because Google uses its well-balanced paths to directly forward inter-cloud traffic from its own ASs to the ASs of the Amazon's and Microsoft's DCs. The traffic from Google's DCs to Alibaba's DCs first traverses the major ISPs in China before entering Alibaba's ASs, but most of the time is spent on traveling Google's own networks, resulting in low latency imbalance. Pairs involving Alibaba's DCs have higher latency imbalance than other pairs mainly because the long-distance paths between Alibaba's and other DCs are not well-balanced. Looking at the AS-level paths between Alibaba's and Amazon's DCs, we found that the high imbalance from Alibaba's to Amazon's DCs largely occurs at the inter-AS connections between AS 4837 (China Unicom) and AS2914 (NTT), while the high imbalance from Amazon's to Alibaba's DCs mainly occurs within AS 4134 (China Telecom), a major ISP in China.

8 APPLICATIONS

As discussed in 6.4.2, it is inherently difficult to guarantee that LB paths have similar latency at the network layer or below. In this section, we will evaluate the impact of latency imbalance on three applications and discuss potential solutions.

8.1 Clock Synchronization by NTP

An NTP client synchronizes time by adjusting its local clock to the NTP server's clock. Suppose that of all LB paths, the i-th path is used for time synchronization with its forward- and return-path latency equal to d_i^F and d_i^R respectively. The error in estimating the server's clock time is equal to the estimation error of the OWD, i.e., $|d_i^F - d_i^R|/2$ [39]. When multiple LB paths exist, choosing forward and return paths that are more symmetric can reduce the estimation error of

the OWD. Suppose A is the set of LB paths. The *maximum* reduction of the estimation error is $\left(\max_{i\in A}\{|d_i^F - d_i^R|\} - \min_{j\in A}\{|d_j^F - d_j^R|\}\right)/2$.

Experiment and Result Summary. We show in Appendix B.1 that the maximum reduction of the estimation error is bounded above by $\overline{\Delta}_d$, where $\overline{\Delta}_d$ denotes the average of one-way imbalance on the forward and return paths. Using the same dataset of OWDs between the 55 nodes (24 data center instances and 31 Planetlab (PL) nodes) in §4.5, our experiments (see Appendix B.2 for details) show that the maximum reduction in synchronization error by considering latency imbalance is larger than $\overline{\Delta}_d/2$ for most source-destination pairs, where $\overline{\Delta}_d$ is the average of one-way imbalance on the forward and return paths. This implies that NTP clients with one-way imbalance larger than the baseline (in Table 1) could have much lower synchronization error if latency imbalance is considered. Among 2671 pairs in our experiments, about 11% of PL-PL pairs, 12% of DC-DC pairs and 21% of DC-PL pairs have one-way imbalance larger than the baseline.

Practical Impact. Although NTP uses DNS-based load balancing to serve clients based on locality, NTP servers could, in practice, still see a median OWD of 175-300ms to worldwide clients [65]. Since long-distance paths are more likely to have high latency imbalance, considering latency imbalance in NTP would benefit a large portion of clients. Moreover, NTP servers could also see traffic from cloud providers [9], and popular IoT clouds (e.g., AWS IoT [66]) would synchronize their servers with cloud-connected IoT devices. The communications between clouds and public Internet would experience similar latency imbalance as characterized in §6.1.

Potential Solution. Using the same dataset, we simulate the schemes of choosing a random path and the lowest-latency path for time synchronization and repeat the experiment 1,000 times. In the experiments, only source-destination pairs with potential improvement ($\overline{\Delta}_d \geq$ 1ms) are used and 10 random flows are used in finding the lowest-latency path. We found that using the lowest-latency path could achieve on average 64% of the maximum error reduction and reduce the standard deviation of synchronization error by 1ms compared to using a random path. However, choosing the lowest-latency path incurs extra overhead. This solution works for any client-server pair, complementary to stratum server selection.

8.2 Delay-Based Geolocation

Delay-based geolocation techniques commonly use `ping` to measure latency and use the measured latency to estimate the distance from landmarks to the host. However, under latency imbalance, `ping` may fail to measure the *minimum path latency* (minRTT), resulting in overestimated distance. We refer to the measured minRTT by `ping` as *ping-time* and the difference between the ping-time and the minRTT as *latency gap*. We show how the accuracy of CBG++ [7], the state-of-the-art geolocation technique, can be improved by considering latency imbalance. Due to limited space, we summarize our results and then present potential solutions. Experimental details are included in the Appendix C.

Experiment and Result Summary. We measure ping-time and the minRTT, in parallel, from DCs in 15 cities around the globe to 400K destinations. We found that the worst DC (with the largest median latency gap) has latency gap > 5ms to 40% of addresses. As landmarks close to the host are found to be more effective in geolocation [7], we further look at DC-address pairs with OWD < 60 ms and find that the distribution of latency gap is similar. This indicates that latency gap occurs at latencies of a wide range.

We simulate delay-based geolocation using the real-world data of 55 PlanetLab nodes. To simulate the process of geolocating an Internet host, we randomly select one PlanetLab node as the host and

use the rest as landmarks. Based on the latencies between landmarks and the host, we use CBG++ to predict a region containing the host. The accuracy of geolocation is measured by the area of the predicted region. Since the predicted region under ping-time is no less than that under the minRTT, we consider the percentage of *area reduction* as the accuracy improvement by considering latency imbalance. We found that using the minRTT can reduce the predicted area more than 20% for 40% of hosts. We further verify if latency imbalance larger than the baseline in Table 1 is significant to geolocation. As latency gap includes latency imbalance on both forward and return paths, we set latency gap to 4ms, equivalent to one-way imbalance equal to 2ms (the baseline). About 85% of hosts have area reduction > 20% both when all landmarks are used or only close landmarks (with OWD < 60ms to the target) are used.

Practical Impact. The results above are obtained using our measured imbalance from the cloud to the public Internet. Applications that possibly see similar improvement include: 1) geolocating VPN services to enforce accurate geographic access control [7] and 2) geolocating data in the cloud to verify that replicas are in diverse geolocations [67] or to prevent storage providers from relocating the data [12]. As imbalanced diamonds exist in the public Internet (§6.4.2), other geolocation applications would also benefit.

Potential Solution. The experiments above used our method to measure the minRTT (see Appendix C.1 for details), where UDP packets with random ports are used to uncover LB paths and the minimum path latency is used as the minRTT. `Ping` uses ICMP echo packets, which have higher reachability than UDP packets [57], but UDP packets may uncover more LB paths than ICMP echo packets as more per-flow load balancers balance UDP packets than ICMP echo packets. Moreover, sending UDP packets with random ports, rather than incrementing ones as traceroute does, helps UDP packets uncover LB paths more efficiently.

8.3 VoIP

Long-distance Internet-based calls have been rising in recent years with 46% of Skype calls being international [13]. However, long-distance Internet paths could have high RTT, packet loss and jitter. Distributed DCs around the globe are envisioned as relays to provide high-quality paths [13]. In this context, we show how to improve call quality when latency imbalance is considered.

Result Summary. Our experiment simulates a relay network formed by 24 DCs around the globe (see Appendix D for details). Our simulated relay network is similar to the ones used by Google+ Hangout and Skype [68], except that we use relay networks to improve performance, not just to provide connectivity between clients. We found that when packet loss is 0.03% (packet loss between DCs [69]), 14% of calls between 10 country pairs comprising 40% of total calls can have better call quality. The average relative imbalance for these calls is > 0.1, confirming latency imbalance larger than the baseline in Table 1 should be significant to VoIP.

Potential Solution. Requests on multiple flows can be sent simultaneously to the relay or destination, and the first received request is considered taking the shortest path. Although we only consider stable path latency, this method is also useful to explore the multi-path capability to reduce latency inflation as in [24].

9 LIMITATIONS

Our methodology and measurement results have several limitations.

Limited View: All of our hosts are in DCs, but ideally we want more heterogeneous ones such as residential hosts. Latency imbalance in public-only Internet may differ from that between the cloud and the public Internet.

Measurement Bias: Our method is biased towards destinations with stable flow latencies. Destinations pruned due to unstable latencies could also have high latency imbalance.
Causes Not Validated: Our tool has limited visibility in MPLS tunnels and has no alias resolution, which prevents us from fully characterizing and understanding the causes for latency imbalance. The possible causes are also not validated by network operators.
Compounding Errors: In §4.3, although we have carefully examined each type of error sources in our methodology, these errors could compound and greatly affect the accuracy of pairwise latency imbalance. For instance, the probabilistic guarantee of high path coverage may fail together with the stability check of path latency during temporary network congestion. If the goal is the accurately measure pairwise latency imbalance, multiple measurements for the same pair over time would help either explore the rare paths with extreme latencies or reduce the measurement errors due to temporary network events.

10 CONCLUSION

In this paper, we developed a methodology of measuring latency imbalance among Internet load-balanced paths at scale and used it to collect a global view of latency imbalance from the perspective of the cloud. We found that latency imbalance is both significant and prevalent between DCs in the cloud and from the cloud to the public Internet. We analyzed the reasons for cloud providers to have different latency imbalance and further showed that latency imbalance could affect application performance in various ways. Moreover, we verified the accuracy of our measured imbalance and discussed the potential causes for latency imbalance and the challenges of load balancing. In the era of 5G, with the last-mile ratio link having sub-millisecond delay and emerging applications being more delay-constrained, the problem of Internet latency imbalance will become more prominent. This requires us to consider latency imbalance in the performance evaluation of future applications.

Our work has limitations in that we measure latency imbalance all from DCs and only focus on latency. As future work, we want to probe from more heterogeneous sources and measure other performance metrics, e.g., throughput, for load balancing. We also want to study diamonds in depth at the Internet scale.

11 ACKNOWLEDGEMENT

We thank all anonymous reviewers for their valuable comments that helped improve our work.

REFERENCES

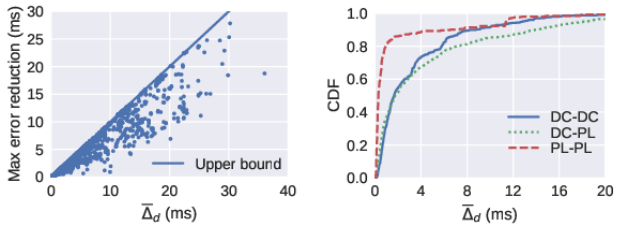
[1] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring load-balanced paths in the Internet. In *IMC*, 2007.
[2] Sandeep Kumar Singh, Tamal Das, and Admela Jukan. A survey on Internet multipath routing and provisioning. *IEEE Communications Surveys & Tutorials*, 17(4):2157–2175, 2015.
[3] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with Paris Traceroute. In *IMC*, 2006.
[4] Kevin Vermeulen, Stephen D. Strowes, Olivier Fourmaux, and Timur Friedman. Multilevel MDA-Lite Paris Traceroute. In *IMC*, 2018.
[5] Imtiaz Parvez, Ali Rahmati, Ismail Guvenc, Arif I. Sarwat, and Huaiyu Dai. A survey on low latency towards 5G: RAN, core network and caching solutions. *IEEE Communications Surveys & Tutorials*, 20(4):3098–3130, 2018.
[6] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving beyond end-to-end path information to optimize CDN performance. In *IMC*, pages 190–201, 2009.
[7] Zachary Weinberg, Shinyoung Cho, Nicolas Christin, Vyas Sekar, and Phillipa Gill. How to catch when proxies lie: Verifying the physical locations of network proxies with active geolocation. In *IMC*, pages 203–217, 2018.
[8] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *SIGCOMM CCR*, pages 15–26, 2004.
[9] Sathiya Kumaran Mani, Ramakrishnan Durairajan, Paul Barford, and Joel Sommers. MNTP: Enhancing time synchronization for mobile devices. In *IMC*, 2016.

[10] Cristina D. Murta, Pedro R. Torres Jr, and Prasant Mohapatra. Qrpp1-4: Characterizing quality of time and topology in a time synchronization network. In *Globecom*, pages 1–5, 2006.
[11] Cristel Pelsser, Luca Cittadini, Stefano Vissicchio, and Randy Bush. From Paris to Tokyo: On the suitability of ping to measure latency. In *IMC*, 2013.
[12] Mark Gondree and Zachary NJ Peterson. Geolocation of data in the cloud. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 25–36, 2013.
[13] Junchen Jiang et al. Via: Improving Internet telephony call quality using predictive relay selection. In *SIGCOMM*, 2016.
[14] Our tool and dataset. https://github.com/yibopi/latency-imbalance.
[15] Yuval Shavitt and Noa Zilberman. A geolocation databases study. *IEEE Journal on Selected Areas in Communications*, 29(10):2044–2056, 2011.
[16] David L. Mills. Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI. https://tools.ietf.org/html/rfc4330, 2006.
[17] Jonathan Ledlie, Paul Gardner, and Margo I. Seltzer. Network coordinates in the wild. In *NSDI*, volume 7, pages 299–311, 2007.
[18] Zhiruo Cao, Zheng Wang, and Ellen Zegura. Performance of hashing-based schemes for Internet load balancing. In *INFOCOM*, pages 332–341, 2000.
[19] Darryl Veitch, Brice Augustin, Renata Teixeira, and Timur Friedman. Failure control in multipath route tracing. In *INFOCOM*, 2009.
[20] Rafael Almeida, Osvaldo Fonseca, Elverton Fazzion, Dorgival Guedes, Wagner Meira, and Ítalo Cunha. A characterization of load balancing on the IPv6 Internet. In *PAM*, 2017.
[21] Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. MPLS under the microscope: Revealing actual transit path diversity. In *IMC*, 2015.
[22] Eric Elena, Jean-Louis Rougier, and Stefano Secci. Characterisation of AS-level path deviations and multipath in Internet routing. In *6th EURO-NGI Conference on Next Generation Internet*, 2010.
[23] Matthew Luckie, Young Hyun, and Bradley Huffaker. Traceroute probe method and forward IP path inference. In *IMC*, 2008.
[24] Zhe Wu, Curtis Yu, and Harsha V. Madhyastha. CosTLO: Cost-effective redundancy for lower latency variance on cloud storage services. In *NSDI*, 2015.
[25] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-based geolocation of Internet hosts. *TON*, 14(6):1219–1232, 2006.
[26] Fangfei Chen, Ramesh K. Sitaraman, and Marcelo Torres. End-user mapping: Next generation request routing for content delivery. In *CCR*, 2015.
[27] Yibo Pi, Sugih Jamin, Peter Danzig, and Jacob Shaha. AP-Atoms: A high-accuracy data-driven client aggregation for global load balancing. *IEEE/ACM Transactions on Networking*, 26(6):2748–2761, December 2018.
[28] Mohammed S. Elbamby, Cristina Perfecto, Mehdi Bennis, and Klaus Doppler. Toward low-latency and ultra-reliable virtual reality. *IEEE Network*, 32(2):78–84, 2018.
[29] Roy D. Yates, Mehrnaz Tavan, Yi Hu, and Dipankar Raychaudhuri. Timely cloud gaming. In *INFOCOM*, 2017.
[30] Zhiruo Cao, Zheng Wang, and Ellen Zegura. Performance of hashing-based schemes for Internet load balancing. In *INFOCOM*, 2000.
[31] Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosenblum, and Amin Vahdat. Exploiting a natural network effect for scalable, fine-grained clock synchronization. In *NSDI*, pages 81–94, 2018.
[32] Ethan Katz-Bassett, Harsha V. Madhyastha, Vijay Kumar Adhikari, Colin Scott, Justine Sherry, Peter Van Wesep, Thomas E. Anderson, and Arvind Krishnamurthy. Reverse traceroute. In *NSDI*, 2010.
[33] MDA-Lite Gitlab repository. https://gitlab.planet-lab.eu/cartography/multilevel-mda-lite, Sep 2018.
[34] Abhinav Pathak, Himabindu Pucha, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao. A measurement study of Internet delay asymmetry. In *PAM*, 2008.
[35] Robert Beverly. Yarrp'ing the Internet: Randomized high-speed active topology discovery. In *IMC*, pages 413–420, 2016.
[36] Perry R. Hinton. Statistics explained. Routledge, 2014.
[37] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.
[38] Youndo Lee and Neil Spring. Identifying and aggregating homogeneous IPv4 /24 blocks with Hobbit. In *IMC*, pages 151–165, 2016.
[39] Ki Suh Lee, Han Wang, Vishal Shrivastav, and Hakim Weatherspoon. Globally synchronized time via datacenter networks. In *SIGCOMM*, 2016.
[40] IP latency statistics. https://enterprise.verizon.com/terms/latency/, July 2019.

[41] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. Salsify: low-latency network video through tighter integration between a video codec and a transport protocol. In *NSDI*, 2018.
[42] CAIDA. Routeviews prefix to AS mappings dataset for IPv4 and IPv6. https://www.caida.org/data/routing/routeviews-prefix2as.xml, Sep 2018.
[43] Timothy Wood, Prashant J. Shenoy, Alexandre Gerber, Jacobus E. van der Merwe, and Kadangode K. Ramakrishnan. The case for enterprise-ready virtual private clouds. In *HotCloud*, 2009.
[44] Sathiya Kumaran Mani, Ramakrishnan Durairajan, Paul Barford, and Joel Sommers. An architecture for IoT clock synchronization. In *Proceedings of the 8th International Conference on the Internet of Things*, 2018.
[45] Alexander Marder, Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, kc claffy, and Jonathan M. Smith. Pushing the boundaries with bdrmapit: Mapping router ownership at Internet scale. In *IMC*, 2018.
[46] CAIDA. The CAIDA Internet Topology Data Kit. https://www.caida.org/data/internet-topology-data-kit, August 2018.
[47] AFRINIC Extended Allocation and Assignment Reports. ftp://ftp.afrinic.net/pub/stats/afrinic/, 2019.
[48] RIPE Extended Allocation and Assignment Reports. ftp://ftp.afrinic.net/pub/stats/ripencc/, 2019.
[49] LACNIC Extended Allocation and Assignment Reports. ftp://ftp.afrinic.net/pub/stats/lacnic/, 2019.
[50] APNIC Extended Allocation and Assignment Reports. ftp://ftp.afrinic.net/pub/stats/apnic/, 2019.
[51] ARIN Extended Allocation and Assignment Reports. ftp://ftp.afrinic.net/pub/stats/arin/, 2019.
[52] Euro-IX IXP Directory. https://www.euro-ix.net/tools/ixp-directory, 2019.
[53] PeeringDB. https://peeringdb.com/api, 2019.
[54] Packet Clearing House: Internet Exchange Directory. https://www.pch.net/applications/ixpdir/menu_download.php, 2019.
[55] CAIDA. The CAIDA AS Relationships Dataset. https://www.caida.org/data/as-relationships/, Feb 2019.
[56] Google cloud network service tiers. https://cloud.google.com/network-tiers#tab1.
[57] Matthew Luckie, Amogh Dhamdhere, kc Claffy, and David Murrell. Measured impact of crooked traceroute. In *CCR*, pages 14–21, 2011.
[58] Ramesh Govindan and Vern Paxson. Estimating router ICMP generation delays. In *PAM*, 2002.
[59] Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. Through the wormhole: Tracking invisible MPLS tunnels. In *IMC*, 2017.
[60] Joel Sommers, Paul Barford, and Brian Eriksson. On the prevalence and characteristics of MPLS deployments in the open Internet. In *IMC*, 2011.
[61] Benoit Donnet, Matthew Luckie, Pascal Mérindol, and Jean-Jacques Pansiot. Revealing MPLS tunnels obscured from traceroute. In *SIGCOMM CCR*, 2012.
[62] Abhinav Pathak, Ming Zhang, Y. Charlie Hu, Ratul Mahajan, and Dave Maltz. Latency inflation with MPLS-based traffic engineering. In *IMC*, 2011.
[63] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven WAN. In *SIGCOMM CCR*, 2013.
[64] Sushant Jain et al. B4: Experience with a globally-deployed software defined WAN. In *SIGCOMM*, pages 3–14, 2013.
[65] Ramakrishnan Durairajan, Sathiya Kumaran Mani, Joel Sommers, and Paul Barford. Time's Forgotten: Using NTP to understand Internet latency. In *HotNets*, 2015.
[66] Amazon AWS IoT. https://aws.amazon.com/iot/.
[67] Karyn Benson, Rafael Dowsley, and Hovav Shacham. Do you know where your cloud files are? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 73–82, 2011.
[68] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. Video telephony for end-consumers: Measurement study of Google+, iChat, and Skype. In *IMC*, pages 371–384, 2012.
[69] Osama Haq, Mamoon Raja, and Fahad R. Dogar. Measuring and improving the reliability of wide-area cloud paths. In *WWW*, 2017.
[70] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *SIGCOMM*, 2001.
[71] Planetlab locations. https://www.planet-lab.org/db/pub/sites.php, April, 2019.
[72] iPlane project dataset. http://web.eecs.umich.edu/~harshavm/iplane/iplane_logs/data/.
[73] G.107. *The E-Model, a computational model for user in transmission planning*, 2017. https://www.itu.int/rec/T-REC-G.107.
[74] Haiyong Xie and Yang Richard Yang. A measurement-based study of the skype peer-to-peer VoIP performance. In *IPTPS*, 2012.

A FIXING THE RETURN PATH

Upon receiving a UDP probe, the destination assembles an ICMP response by appending the IP and UDP headers of the UDP probe after the ICMP header. The ICMP checksum is computed based



(a) Max error reduction (b) Average one-way imbalance

Fig. 11. Maximum error reduction for clock offset in NTP

on the ICMP header and the appended data. Except for the checksum field (cleared to 0 when the checksum is cacluated), other fields in the ICMP headers are the same for responses to the same address. The ICMP checksum thus is determined by the appended data. In the appended data, the IP headers are the same for UDP probes sent from our probe to the same destination, while the UDP headers use different port numbers for each flow. The variability of port numbers can be offset by the UDP checksum if we send UDP probes with no UDP data. In this case, the one's complement sums of the UDP headers are the same, or equivalently, the checksums of ICMP responses are the same for different flows to the same address.

B TIME SYNCHRONIZATION BY NTP

B.1 Upper Bound of Error Reduction

Suppose there are n LB paths between the NTP client and the NTP server and that the maximum and minimum estimation errors occur at the i-th and j-th LB paths respectively. The maximum reduction of estimation error (denoted as Δ) then becomes $\left(|d_i^F - d_i^R| - |d_j^F - d_j^R|\right)/2$. Using triangle inequality, we have that

$$\begin{aligned}\Delta &\le \left|(d_i^F - d_i^R) - (d_j^F - d_j^R)\right|/2 \le \left(|d_i^F - d_j^F| + |d_i^R - d_j^R|\right)/2 \\ &\le (\Delta_F + \Delta_R)/2,\end{aligned}$$

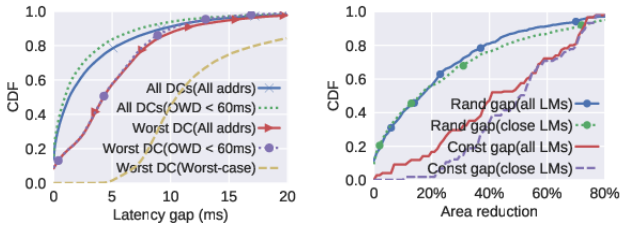
where Δ_F and Δ_R are the one-way imbalance on the forward and return paths respectively.

B.2 Relation Between One-Way Imbalance and NTP Accuracy

Using the same dataset of OWDs between the 55 nodes (24 data center instances and 31 Planetlab (PL) nodes) in §4.5, we calculate the maximum error reduction for each pair of nodes as in §8.1. Figure 11(a) shows the relation between the maximum error reduction and $\overline{\Delta}_d$ for all pairs of nodes. The maximum error reduction can be as high as 27ms and more importantly, most of the maximum error reductions are larger than $\overline{\Delta}_d/2$. Figure 11(b) shows the CDF of the average latency imbalance of different types of pairs. The latency imbalance between PL-PL pairs is much lower than that between pairs involving DCs.

C DELAY-BASED GEOLOCATION

We first present a global view of latency gap and then discuss how latency gap affects geolocation accuracy. The global view of latency gap is from the perspective of DCs, providing insights to the geolocation for cloud-based applications, e.g., location verification of cloud data and proxies [7, 12].



(a) Distribution of latency gap (b) Distribution of area reduction

Fig. 12. Impact of latency imbalance on geolocation

C.1 A Global View of Latency Gap

We measure ping-time and the minRTT, in parallel, from DCs in 15 cities around the globe to 400K random destinations. We use Amazon's and Alibaba's DCs as vantage points to reflect more closely traffic sent from client hosts in the Internet (see § 6.2). For each DC-address pair, we repeat the method in §4.3 with UDP, TCP and ICMP probes respectively to discover as many alternate paths as possible and use the minimum RTT sample as minRTT. We use `fping`, an extension to `ping`, for fast probing destinations in parallel. In geolocation applications, *ping-time* is typically the minimum of several RTT samples to mitigate queueing delay [25, 70]. In our experiments, we use more samples (i.e., 30 samples) to further exclude queueing delay for *ping-time*. Figure 12 shows the distribution of latency gaps under different cases. The worst DC (with the largest median latency gap) has latency gap > 5ms to 40% of addresses. As landmarks close to the host are found to be more effective in geolocation [7], we further look at DC-address pairs with OWD < 60 ms and find that the distribution of latency gap is similar. This indicates that latency gap occurs at latencies of a wide range. The worst case assumes that `ping` probes sent from the worst DC always take the highest-latency path, indicating the maximum possible latency gap. The figure shows that the worst-case latency gap is much larger than those under other cases.

C.2 Impact of Latency Gap on Geolocation Accuracy

We simulate delay-based geolocation using the real-world data of 55 PlanetLab nodes, where their locations can be found in [71] and the RTT samples between them are obtained from the iPlane project dataset [72]. To simulate the process of geolocating an Internet host, we randomly select one PlanetLab node as the host and use the rest as landmarks. Based on the latencies between landmarks and the host, we use CBG++ to predict a region containing the host. To study the impact of latency gap, the region is predicted under both minRTT (without latency gap) and ping-time (with latency gap). The minRTT between each landmark and the host is the minimum RTT sample between them in the iPlane project dataset and the ping-time is simulated as the minRTT plus latency gap. We use the DC-address pairs in Figure 12(a) to resemble the landmark-host pairs in applications using existing measurement platforms to geolocate cloud addresses, where the host resides in a DC and landmarks are common Internet addresses outside the cloud. We consider two geolocation schemes: 1) all landmarks are used for geolocation, where the latency gap follows the distribution when all DC-address pairs are included in Figure 12(a) and 2) only landmarks close to the host (with OWD < 60) are used for geolocation, where the latency gap follows the distribution when only DC-address pairs with OWD < 60 are included.

The accuracy of geolocation is typically measured by the area of the predicted region. Since the predicted region under ping-time is no less than that under minRTT, we consider the percentage of *area reduction* as the accuracy improvement by considering latency imbalance. We repeat the geolocation process above for 1000 random hosts (selected with replacement from the 55 PlanetLab

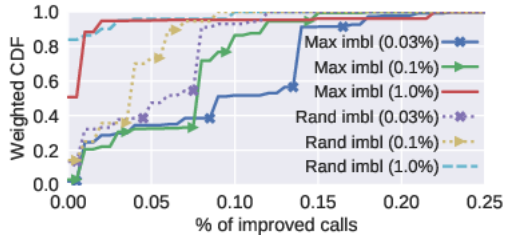


Fig. 13. Weighted distribution of the percentage of improved calls for country pairs

nodes) under each geolocation scheme. Figure 12(b) shows the distribution of area reduction (in percentage), where area reduction is > 20% for 40% of hosts. The area reduction would be larger if the host is in the worst DC. We further verify if latency imbalance larger than the baseline in Table 1 is significant to geolocation. As latency gap includes latency imbalance on both forward and return paths, we set latency gap to 4ms for each landmark-host pair, equivalent to one-way imbalance equal to 2ms (the baseline). About 85% of hosts have area reduction > 20% for both geolocation schemes.

D VOIP

Our experiment simulates a relay network formed by 24 DCs around the globe, where a path between the caller and callee consists of multiple *logical links* (including a caller-relay link, a relay-callee link and relay-relay links, if used) and each link could include multiple alternate paths. Our simulated relay network is similar to the ones used by Google+ Hangout and Skype [68], except that we use relay networks to improve performance, not just to provide connectivity between clients. The *best path* is defined to have the minimum latency between the caller and callee. Without considering latency imbalance, each link randomly uses one of the alternate paths. When latency imbalance is considered, each link always uses the lowest-latency path. We select the best paths in both cases and consider the difference between call quality as the improvement by considering latency imbalance. The call quality is estimated using an analytical model (E-Model defined by the ITU [73]) that computes the Mean Opinion Score (MOS) from network metrics, where MOS values range from 1 ("unacceptable") to 5 ("excellent"). Since our focus is on the impact of latency imbalance on call quality, we fix the jitter parameter in the E-model to the default value [74] and assume constant packet loss for all paths.

We randomly select callers and callers in different countries for each call and consider that call quality is improved if its MOS increases by one or more levels. We simulate one million calls in total with three packet loss rates (0.03%, 0.1% and 1%) and two types of latency imbalance: 1) *random imbalance*, where an alternate path is randomly selected for each link, and 2) *maximum imbalance*, where the highest-latency path is selected for each link. We group calls from the same pair of countries (*country pair* henceforth) and calculate the percentage of calls with improved quality for each country pair. Figure 13 shows the distribution of the percentage of improved calls weighted by the number of calls in respective country pairs. The percentage of improved calls decreases as the packet loss increases and becomes the major bottleneck. Nonetheless, calls between some countries still benefit significantly from latency reduction. When packet loss is 0.03% (packet loss between DCs [69]), the percentage of improved calls is about 8% under random imbalance and 14% under maximum imbalance for 10 country pairs comprising 40% of total calls.

E DETERMINING SAMPLE SIZE FOR MINIMUM PATH LATENCY

We consider one RTT sample as the minimum path latency if it remains the minimum for n consecutive rounds of probing. To determine n, we collect 100 RTT samples of a flow from a DC

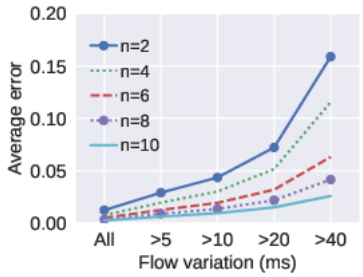


Fig. 14. Choosing sample size for minimum path latency

to each of 100K destinations and use these RTT samples to simulate the process of determining the flow latency in §4.3 under different n's and different network conditions. For each flow, the minimum of its 100 RTT samples is considered its minimum flow latency, d_{min}, and the measured minimum is d_{meas}. We then calculate the error as $|d_{meas} - d_{min}|/d_{min}$ for each flow. Figure 14 shows the average error under different n's and different flow variations, where the variation of a flow is the standard deviation of all its RTT samples and flow variation $> x$ means that only flows with standard deviation of samples $> x$ are used. The average error increases with flow variation. When flow variation is $>$ 40ms, the average error reaches 0.06 when $n = 6$. In our dataset, only 2.6% of flows have variation $>$ 40ms. We choose n to be 6 as a good tradeoff between accuracy and measurement overhead.

Received January 2020; revised February 2020; accepted March 2020