Internet Performance from Facebook's Edge*

Brandon Schlinker[†♯] Italo Cunha[‡♮] Yi-Ching Chiu[†] Srikanth Sundaresan[♯] Ethan Katz-Bassett[♮]

[†] University of Southern California [♯] Facebook [‡] Universidade Federal de Minas Gerais [♮] Columbia University

ABSTRACT

We examine the current state of user network performance and opportunities to improve it from the vantage point of Facebook, a global content provider. Facebook serves over 2 billion users distributed around the world using a network of PoPs and interconnections spread across 6 continents. In this paper, we execute a large-scale, 10-day measurement study of metrics at the TCP and HTTP layers for production user traffic at all of Facebook's PoPs worldwide, collecting performance measurements for hundreds of trillions of sampled HTTP sessions. We discuss our approach to collecting and analyzing measurements, including a novel approach to characterizing user achievable goodput from the server side. We find that most user sessions have MinRTT less than 39ms and can support HD video. We investigate if it is possible to improve performance by incorporating performance information into Facebook's routing decisions; we find that default routing by Facebook is largely optimal. To our knowledge, our measurement study is the first characterization of user performance on today's Internet from the vantage point of a global content provider.

CCS CONCEPTS

• **Networks → Network performance analysis**; *Transport protocols*; *Network performance modeling*; *Network monitoring*.

ACM Reference Format:

Brandon Schlinker, Italo Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. 2019. Internet Performance from Facebook's Edge. In *Internet Measurement Conference (IMC '19), October 21–23, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3355369.3355567

1 INTRODUCTION

Global content providers and content delivery networks have built points of presence (PoPs) around the world to provide short, direct paths to users, with the end-goal of improving user experience [15, 17, 26, 43, 49, 55, 68]. Given the large investment in this infrastructure, it is natural to ask whether it has resulted in users experiencing good performance.

While an extensive body of prior work has investigated user performance on the Internet across a variety of dimensions, many of these studies have focused on a specific region [25, 36], access link type [9, 70], or aspect of the Internet ecosystem that relates to performance such as traffic engineering at interconnections [3, 5, 36, 55]. Studies that attempted to more broadly characterize user performance or opportunities for performance-aware routing have been limited in vantage points and measurements, limiting the analysis and conclusions they could make [3, 5–7, 18, 55, 63].

In this work we characterize the traffic properties and network performance experienced by users of Facebook, a content provider with over 2 billion users across hundreds of countries. In contrast to previous work, we use a dataset of user traffic collected at all of Facebook's PoPs worldwide (§2.2), a subset of Facebook's edge serving infrastructure. Our 10 day dataset is composed of metrics from randomly sampled production traffic, captures performance for hundreds of trillions of HTTP sessions, and has global coverage with measurements from hundreds of countries and billions of unique client IP addresses. The dataset provides the high-volume of samples required to conduct granular analysis of performance, such as identifying spatial and temporal variations. Given this coverage, and because a large share of global Internet traffic comes from a small number of well connected content and cloud providers with connectivity similar to Facebook's [26, 54], performance measurements and analysis from Facebook's vantage point likely roughly reflect user performance to popular services in general. In addition, the dataset contains measurements from production traffic continuously routed via alternate routes. We explore if Facebook's extensive connectivity creates opportunities for performance-aware routing, in which an alternate (non-default) route offers better performance.

We begin with a characterization of Facebook's user traffic, which is predominantly composed of TCP flows carrying HTTP/1.1 or HTTP/2 traffic (§2.3). Most objects requested by users are small (50% of objects fetched are less than 3 KB), and HTTP sessions can be idle for the majority of their lifetimes.

We quantify performance by capturing latency and goodput measurements from existing production traffic (§§ 2.2 and 3). Latency is important in interactive applications where a user is actively blocked awaiting a response, such as waiting for a search query to return or a video to start playing. Goodput depends on latency and other connection properties such as loss, jitter, the congestion control algorithm, and the available bandwidth, and high goodput is important for streaming high-quality video and for downloading large objects. However, capturing insights into achievable goodput from passive traffic measurements is challenging given that most objects served by Facebook are small; under such conditions goodput measurements often under-utilize the available bandwidth as they may not exercise the bandwidth-delay product and/or because of TCP slow-start. Our novel approach to goodput measurements accounts for these intricacies by determining (1) if transmission

*In this paper, Facebook's edge is the collection of PoPs on Facebook's AS.

†Part of this work was done while Yi-Ching Chiu was an intern at Facebook.

IMC '19, October 21–23, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-6948-0/19/10…$15.00
https://doi.org/10.1145/3355369.3355567

of an object is capable of testing for a given goodput (we focus on 2.5Mbps, the minimum bitrate for HD video) and (2) for capable transmissions, if the transmission *achieved* the target goodput after correcting for transmission time, CWND growth, and other aspects (§3.2). This approach enables us to differentiate between goodput restricted by network conditions (which we want to measure) and goodput "only" restricted by sender behavior. Our approach is practical and deployed in production at Facebook's PoPs worldwide.

Using the results from our methodology, we characterize the performance seen by our users worldwide (§4). We show that a majority of user sessions have low latency (median MinRTT < 40ms) and achieve the goodput required to stream HD video. We examine regional variances and show that users in Africa, Asia, and South America in particular experience poorer performance.

We aggregate measurements by geolocation information and time to facilitate spatial and temporal analysis, and employ statistical tools when comparing aggregations to separate measurement noise from statistically significant differences. We identify episodes of performance degradation that could be caused by failures and predictable periods of degradation that could indicate recurring downstream congestion. However, the vast majority of traffic sees minimal degradation over the 10 days in the study period (§5).

Finally, we investigate whether it is possible to improve user performance by changing the route Facebook uses to send traffic to users. Prior publications on SDN egress controllers such as Facebook's Edge Fabric [55] and Google's Espresso [68] discuss opportunities for incorporating real-time performance measurements, but did not define a concrete methodology for capturing and converting metrics into decisions and did not fully quantify the benefits of performance aware routing. We offer insights into both of these aspects and find that there is limited opportunity for benefit (§6). Performance-aware routing decisions provide a latency improvement of 5ms or more for only a few percent of traffic of traffic, showing that the existing standard, static BGP routing policy employed by Facebook is close to optimal.

2 DATA COLLECTION OVERVIEW AND TRAFFIC CHARACTERISTICS

This section presents an overview of the Facebook content serving infrastructure as it pertains to serving client traffic (§2.1). We describe our passive measurement infrastructure and the data that we collect (§2.2) and then explore the characteristics of client connections at the application and transport layers (§2.3). In Section 3, we use these insights to illustrate the challenges to quantifying user performance from passive measurements in this environment and how our approach overcomes these challenges.

2.1 Background

The vast majority of Facebook's user traffic is HTTP/1.1 or HTTP/2 secured with TLS atop a TCP transport, which we refer to as an HTTP *session*. A client establishes an HTTP session with an HTTP *endpoint* (distinguished by IP address) depending on the application and the type of request. Each HTTP session can have one or more *transactions*, each composed of an HTTP request from the client and response from the server.

Proxygen, Facebook's software load balancer, runs at each endpoint, terminates client TCP connections, and forwards HTTP requests to internal services [57]. Throughout this work, when we refer to a *server* we are referring to a Proxygen load balancer.

Facebook has dozens of PoPs across six continents and a significant fraction of client traffic is directed to endpoints at these PoPs; our analysis is centered around this traffic. Most users are close to a PoP; based on geolocation of users, half of all traffic is to users within 500km of the serving PoP, and 90% is to users within 2500km and in the same continent. The 10% of traffic served by a PoP in a different continent than the user is composed predominantly of European PoPs serving users in Asia (4.8% of all traffic) and Africa (2.1% of all traffic).

Two complementary traffic engineering systems are responsible for steering user traffic. Cartographer steers client traffic to PoPs by controlling DNS responses and rewriting endpoint URLs in HTML, using performance measurements to decide the best ingress location [56]. Egress traffic at a PoP is routed based on decisions are made by Edge Fabric [55], which we discuss in further detail in Section 6.

2.2 Measurement Infrastructure and Dataset

We measure performance by collecting metrics from existing production traffic at our servers. We present the reasoning behind this design choice and its trade-offs relative to approaches used in prior work. We also describe how we collect data and the properties of the dataset.

2.2.1 Why server-side passive measurements? Prior work has characterized performance via instrumentation at clients that executes measurement tasks such as downloading an object or pinging an endpoint [3, 5, 19] and/or network probing measurements executed from the server side, such as pings or traceroutes [71].

In contrast, we use server-side measurements of existing production traffic as they best support our measurement goals:

Avoid overhead at clients: Capturing measurements at the server side from existing traffic does not introduce any overhead at clients. In comparison, executing measurements from clients must be done with extreme care to avoid negative consequences. For instance, measuring goodput may require transferring a large volume of traffic (§3.2) and thus may increase data usage and reduce battery life on mobile devices.

Facilitate granular time series analysis: Server side measurements can satisfy the high sampling rates required for temporal analysis. Events such as congestion or failures can cause network performance to quickly change. Detecting such events and evaluating options to mitigate requires sufficient samples to make statistically significant conclusions at short-time scales. Capturing sufficient samples with active measurements is challenging [64].

Ensure representative results: Capturing measurements from existing production traffic ensures that results are representative of user performance. In comparison, measurements collected server-side via pings or traceroutes may not be representative. ICMP traffic may be deprioritized, dropped, or routed over a different path than TCP traffic [41, 59, 60], and prior work has found that over 40% of

hosts do not respond to ICMP probes, limiting the coverage of systems that rely on such probes [41, 71]. In the same vein, experiments executed on randomly selected production flows are guaranteed to not be biased by the sampling scheme.

Enable rapid experimentation: Changes at the server side are easy and quick to roll out and maintain, in direct contrast with client changes which typically have longer rollout cycles and require extensive testing. For instance, in this work we evaluate whether changes in how we route traffic to users can improve performance (§6) without requiring any coordination with clients.

However, server side measurements have two key drawbacks:

May not capture end-to-end performance: Performance enhancing proxies (PEPs) are commonly deployed in satellite and cellular networks and attempt to improve performance by splitting the TCP connection between the user and server and then optimizing TCP behavior for each segment's characteristics [14, 67]. Under these conditions, server side performance measurements reflect the performance between Facebook and the PEP instead of end-to-end performance, and thus may overestimate goodput and underestimate latency relative to what would be measured end-to-end.[1] However, since Facebook can only optimize for conditions between Facebook's edge and the PEP, this does not have any considerable drawback on our analysis.

Experiments can degrade performance: Experiments that impact production traffic could (inadvertently) degrade performance for users. Limiting the traffic impacted by an experiment reduces this risk, but we acknowledge that this may be insufficient. Prior work has cited risks to production traffic in justifying separate active measurement infrastructure for experiments [19].

2.2.2 Measurement Infrastructure. We capture performance information at the load balancers, which are configured to sample a percentage of HTTP sessions. For sampled sessions, instrumentation captures TCP state at the start and end of each HTTP session, and for each HTTP transaction, the load balancer captures timestamps and TCP state at prescribed points to enable calculation of goodput. We discuss how we use this state further in Section 3.

When the load balancer detects a sampled session's underlying TCP connection has been closed, it captures the final TCP state. The load balancer then sends all captured information to a separate process that adds information about the egress route used to deliver traffic to the user, including BGP information such as the destination prefix and AS-path.

2.2.3 Control of egress routes. Traffic to a given destination may be shifted to an alternate route by Edge Fabric, Facebook's egress routing controller, if an interconnection at the edge of Facebook's network is at risk of becoming congested [55]. These shifts could impact performance for the small fraction of traffic that is shifted and would complicate analysis of opportunity for performance aware routing. To ensure that our analysis is not influenced, servers override the route used for sampled HTTP sessions in coordination with Edge Fabric. The analysis in Sections 4 and 5 always uses measurements from the best route according to Facebook's routing policy (§6.1). Likewise, the analysis in Section 6 always compares the best route against n alternate routes. This ensures that our analysis is not impacted by Edge Fabric's actions or congestion at the edge of Facebook's network.

2.2.4 Dataset. All of our analysis is derived from a dataset containing 10 days worth of samples collected from load balancers at all of Facebook's PoPs worldwide. We describe the metrics each sample captures in Section 3. Servers randomly select HTTP sessions to sample at a defined rate, and a subset of sessions are routed via alternate egress routes to enable comparisons of performance across routes (§6). Since our focus is on performance between Facebook's edge and users, we filter samples to client IP addresses determined by a third-party commercial service to be controlled by a hosting provider (~2% of measured traffic).[2]

Post filtering, our dataset contains measurements from billions of unique client IP addresses spread across hundreds of countries. In total, our dataset contains measurements for hundreds of trillions of HTTP sessions.

2.3 Connection Characteristics

In order to inform the design of our performance measurements, we characterize Facebook's user traffic at the HTTP session and transaction level. We show that the majority of session time (e.g., the time from establishment to termination of the underlying TCP connection) is spent idle, and sessions and transactions transfer small amounts of data. Section 3 discusses how these insights shape our approach to measuring goodput.

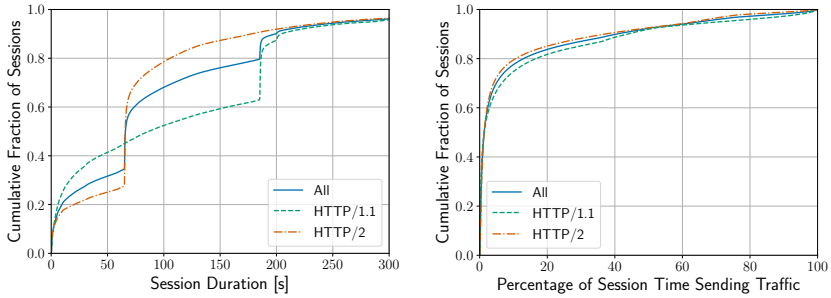
HTTP Sessions are mostly idle. Figure 1(a) shows the distribution of client session durations. Session times vary, with 7.4% lasting for less than a second and 33% lasting for less than a minute, and 20% lasting more than 3 minutes. HTTP/2 sessions, which are commonly used by web browsers and some of Facebook's mobile applications, have fewer short sessions than HTTP/1.1. For example, 44% of HTTP/1.1 sessions lasted for less than a minute, while only 26% of HTTP/2 sessions did. Figure 1(b) shows the percentage of time that the load balancer is actively sending data for an HTTP session (i.e., the load balancer has data to send to the client and/or there is unacknowledged data in flight). For both HTTP/1.1 and HTTP/2, the majority of sessions are idle for most of their lifetime; 80% of HTTP/2 sessions are active less than 10% of the time, while 75% of HTTP/1.1 sessions are active less than 10% of the time.

A small fraction of sessions transfer the bulk of data volume. Figure 2 shows the distribution of the number of bytes transferred per HTTP session. Over 58% of sessions transfer fewer than 10 kilobytes. However, for the subset of sessions to endpoints that host image and video content, 17% of responses transfer at least 100 kilobytes. In addition, there is a long tail with 6% of sessions transferring over 1 megabyte, the majority of which contain transactions for streaming video objects.

Figure 2 also shows the distribution of response sizes across transactions. Over 50% of responses are fewer than 6 kilobytes, the

[1] Adoption of QUIC will nullify this drawback as QUIC's encryption inherently prevents PEPs from splitting connections [42, 45, 66].

[2] We have found that these HTTP sessions are composed of API requests (from other organization's servers) and traffic relayed by VPN providers. VPN traffic can mislead temporal performance analysis because the composition of users and user locations behind the IP address sourcing VPN traffic can change drastically over time, which in turn changes performance.



(a) Connection duration (b) Connection busy time

Figure 1: Cumulative distribution across sessions of duration and busy times. Most sessions end within 60 seconds and spend most of their lifetime idle.

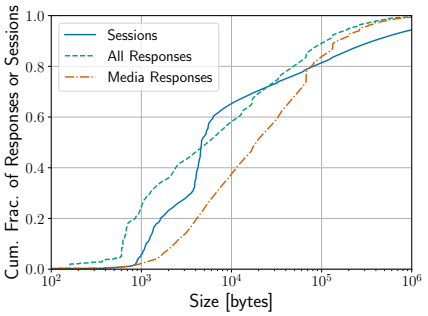


Figure 2: Distribution of bytes transferred per session, all HTTP responses, and media responses.

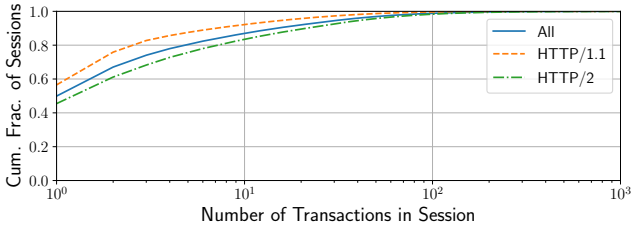


Figure 3: Over 80% of sessions have fewer than 5 transactions, but the majority of traffic is on sessions with greater than 50 transactions (not shown).

vast majority of which are responses to API calls, rendered HTML, and other dynamically generated content. Endpoints hosting images and videos have a slightly larger response size, with a median response size of ~19 kilobytes.

HTTP sessions comprise a small number of transactions. Figure 3 shows that most sessions have a single transaction, and over 87% of HTTP/1.1 and 75% of HTTP/2 sessions have fewer than 5 transactions. For HTTP/1.1, web browsers may establish multiple connections to the same endpoint to facilitate pipelining and enable multiple objects to be requested in parallel. In comparison, multiplexing and pipelining are supported within an HTTP/2 connection, so web browsers only establish a single HTTP/2 connection to an endpoint. As a result, HTTP/2 sessions have more transactions than HTTP/1.1 on average. However, the bulk of total traffic is carried by sessions with many transactions. Sessions with 50 or more transactions account for more than half of all network traffic.

3 QUANTIFYING PERFORMANCE

Our main goal in this section to provide insights into how network quality varies across users, and in later sections across time and routes (§§ 5 and 6). We therefore design our methodology to be agnostic to the specific application workload or client device type.

Although requirements for good performance vary by application—for instance, streaming video traffic requires high goodput and has soft real-time latency demands [29] while real-time games exchange data at low rates but require low latency [40]—latency, loss, jitter, and goodput form baseline metrics commonly considered for assessing network quality. Techniques to capture these metrics have been incorporated into network debugging tools that rely on active measurements [1, 22]. Our methodology for quantifying performance focuses on capturing latency and goodput, with goodput also offering insights into loss and jitter. Our methodology differs from that used in network debugging tools because we seek to extract insights by measuring existing production traffic (§2.2.1).

3.1 Measuring latency with MinRTT

Latency is important in interactive applications where a user is actively blocked on a response, such as waiting for a search query to return or a video to start playing. Prior work has quantified the impact of large changes in latency (100ms+) on e-commerce and other applications [11, 47] and provides rules of thumb that we can use to assess the quality of experience given different latencies.

- Prior research has established that beyond about 8Mbps, latency is the primary bottleneck for page load times in last-mile access networks [65].
- An online gaming services provider uses 80ms latency as a cutoff for good performance [48].
- ITU-T G.114 recommends at most a 150ms one-way delay (300ms RTT) for telecommunications applications; higher latencies significantly degrade user experience [2].

We measure latency by capturing the minimum round-trip time (MinRTT) as measured by the Linux kernel's TCP implementation at connection termination. This metric captures the minimum round-trip time observed over a configurable window; in Facebook's environment this window is set to 5 minutes. Because the vast majority of HTTP sessions terminate within 5 minutes (§2.3), recording this metric at session termination effectively captures the minimum RTT observed over the session's lifetime.

We use MinRTT because it represents an upper bound on the underlying path's propagation delay. Congestion in the backbone will result in a persistent standing queue [28] that increases the MinRTT to be more than the propagation delay (and also can cause loss, which we capture with our goodput metric (§3.2)). In Section 4 we show distributions of user latency from Facebook's perspective based on MinRTT.

Measured RTT can vary over a session's lifetime, and the kernel maintains an RTT calculation that incorporates these variations (sRTT) for calculating the retransmission timeout (RTO).

However, because our focus is the quality of the routes between Facebook and users and *not* the quality of last mile access links or customer premise networks, variation in latency (and more generally jitter) is less relevant to our analysis. Prior work has found that traffic arrival rates in backbone networks at small time scales are smooth because small variations in round-trip time and processing time desynchronize the large number of flows, and flow transfer rates are slow relative to backbone link capacity [32, 53]. As a result, we expect variations in RTT can often be attributed to last mile/customer premise network conditions, such as on-path buffers at the access link becoming bloated due to self-induced, non-standing congestion [34, 62], or wireless/cellular signal quality issues causing retransmissions at the link layer.

3.2 Measuring goodput with HDratio

User experience depends on Facebook's ability to deliver content to users in a performant manner, which in part depends on goodput. For instance, after a video has started playing, user experience is primarily dependent on the connection's ability to sustain the playback bitrate [29]. Clients with low goodput will be unable to continuously stream high bitrate video without frequent rebuffering and will experience delays in loading of images and other large objects. Clients with extremely low goodput may be unable to interact with Facebook's services because requests may timeout before delivery has been completed.

Goodput depends on latency and other connection properties such as loss, jitter, the congestion control algorithm, and the available bandwidth. Individually, these signals do not provide a complete picture. For example, MinRTT and sRTT do not reflect the impact of loss, which impacts data delivery latency and responsiveness at the application level [60], and the frequency and impact of loss depends (among other things) on the congestion control algorithm, the bottleneck link speed, and latency [20, 38].

Goodput reflects the complex interactions between these components and provides a signal that we can use to define the quality of routes between Facebook and users. As with latency, goodput depends on conditions and properties of the backbone, access, and customer premise networks. A client may have low goodput because of limitations in the client's Internet plan or access technology; insight into such limitations can be informative during the development of Facebook's applications and services, but is not actionable in terms of changes to Facebook's edge network. However, goodput measurements provide a powerful tool for temporal analysis of degradation and opportunities for performance-aware routing, both of which may be actionable. For instance, if two routes to a destination network have similar RTTs, but one path has a lower goodput (perhaps due to loss caused by congestion), shifting traffic to the route with higher goodput may improve client performance.

3.2.1 Defining target goodput. Speedtests are often used to capture the maximum achievable goodput between a server and user, but this *maximum* does not correlate linearly with application performance. A client that can achieve 100 Mbps goodput will rarely experience any improvement in performance over a client with 10 Mbps goodput because neither connection is likely to be saturated by the types of content served by Facebook (voice or video calls, live or time-shifted streaming video, and photos) [65]. In addition, speedtests can be intrusive and can have a negative impact on users, and thus are not suitable to our setting (§2.2).

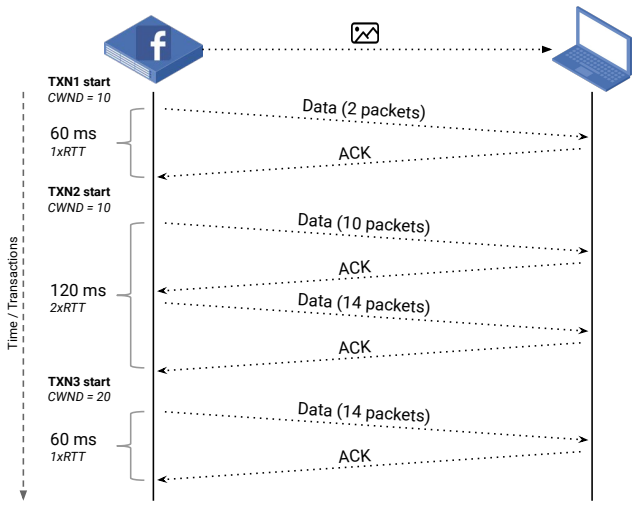


Figure 4: Sequence diagram for three back to back HTTP transactions over a single HTTP session.

Instead of capturing maximum achievable goodput, we define a target goodput that suffices to provide good experience for Facebook's services, and then design our methodology to check whether sessions are able to achieve this goodput. Given the importance of video on today's Internet, we define our target goodput as 2.5 Mbps, the minimum required to stream HD video [4]. We refer to this target goodput as *HD goodput*. Once HD goodput is achieved, latency (which determines responsiveness) may be more important. Although we focus on HD goodput, our methodology is generic and can work for any target goodput.

Because Facebook HTTP sessions are idle for most of their duration (§2.3), goodput at the session level is not meaningful; we must calculate goodput at the transaction level. However, even at the transaction level, capturing meaningful goodput measurements from small responses is non-trivial compared to a speedtest. With small responses, goodput can be restricted by response size or the CWND at the start of a response (rather than by underlying network conditions), and transmission time can have a significant impact.

In the sections that follow, we describe how we determine if an HTTP transaction is capable of *testing* for our target HD goodput (§3.2.2). Then, for the subset of transactions capable of testing, we describe how we determine if the transaction *achieved* the target HD goodput while correcting for transmission time and other aspects (§3.2.3). Finally, we describe how we summarize these results into *HDratio*, a metric that reflects the ability of a given HTTP session to deliver traffic at a rate sufficient to maintain HD goodput (§3.2.4).

3.2.2 Determining if a transaction tests for target goodput. In Facebook's environment, a transaction's response size is small relative to the amount of data transferred in a speedtest (§2.3). The example scenario in Figure 4 illustrates how small responses impact the design of our approach to measure goodput. In this example, a client with a 60ms MinRTT requests three objects in series via a single

HTTP session. We assume *ideal network conditions*, under which a connection experiences fixed round-trip times and no packet losses. To keep the example simple, we (1) assume that there is no bottleneck in the network; (2) ignore the impact of pacing, transmission delay, and ACK clocking; (3) set the maximum packet size to 1500 bytes and the initial congestion window (CWND) to 10 packets; and (4) assume a loss-based congestion control such as Reno. With no loss and fixed round-trip times, the connection will never exit slow start, and the CWND will grow exponentially whenever the connection is *CWND limited*.[3] Under these conditions, goodput is restricted by response size and the CWND at the start of a response (denoted W_{start}).

- Transaction 1: Goodput = 0.4Mbps (2 packets/60ms). W_{start} = 10, greater than response size, takes 1 RTT to send all packets and receive ACK. No CWND growth.
- Transaction 2: Goodput = 2.4Mbps (24 packets/120ms). W_{start} = 10, less than response size, takes 2 RTT to send all packets and receive ACK. CWND grows to at least 20.
- Transaction 3: Goodput = 2.8Mbps (14 packets/60ms). W_{start} = 20, greater than response size, takes 1 RTT to send all packets and receive ACK.

We make three observations from this example:

Goodput can be calculated per RTT. When a transaction's response is transferred over multiple RTT (*i.e.,* W_{start} is less than response size), goodput for a single RTT can be greater than goodput for the transaction. For instance, transaction 2 sends 10 packets in its first RTT, yielding a goodput of 2Mbps, and 14 packets in its second RTT, yielding a goodput of 2.8Mbps.

Goodput per RTT depends on prior transactions. The CWND at the start of a response restricts the goodput a transaction can achieve and depends on previous transactions. For instance, transaction 3 is able to transfer 14 packets in its first and only RTT because transaction 2 grew the CWND to 20. In comparison, transaction 1 did not grow the CWND, so transaction 2 could only transfer 10 packets in its first RTT.

The highest per-RTT goodput under ideal network conditions is the highest goodput a transaction can test for. This type of ideal-case analysis determines the maximum achievable goodput in an RTT for a given W_{start} and response size, providing an upper bound on the highest goodput the response can exercise under real network conditions. From the example, we can observe that the maximum testable goodput for a given response is the maximum number of bytes delivered in a single round-trip:

- Transaction 1 can test if 0.4Mbps goodput can be achieved.
- Transaction 2 (because of its second RTT) and transaction 3 can test whether a goodput of 2.8Mbps can be achieved.

In production we determine the maximum goodput that each transaction can test for by modeling TCP's behavior under ideal conditions. The maximum testable goodput occurs either on the last round-trip or the penultimate round-trip if the last round-trip has fewer bytes to send.

We define m as the number of round-trips required to transfer a response of B_{total} bytes if the CWND size (in bytes) when the first byte of the transaction was sent is W_{start} (note that this may *not* be the same as the initial congestion window, as we explain below):

$$m = \lceil \log_2(B_{\text{total}}/W_{\text{start}} + 1) \rceil. \quad (1)$$

We define $W_{\text{SS}}(n)$ as the size of CWND at the start of the n^{th} round-trip:

$$W_{\text{SS}}(n) = 2^{(n-1)} \times W_{\text{start}}. \quad (2)$$

The maximum testable goodput is the maximum number of bytes transferred over each of the last two round-trips divided by the RTT:

$$G_{\text{testable}} = \frac{\max\{W_{\text{SS}}(m-1), B_{\text{total}} - \sum_{i=1}^{m-1} W_{\text{SS}}(i)\}}{\text{MinRTT}}. \quad (3)$$

In Figure 4, transaction 2 has $m = 2$, $W_{\text{SS}}(2) = 20$, and G_{testable} = 2.8Mbps.

Thus, if a transaction's G_{testable} is greater than or equal to HD goodput, then the transaction is capable of testing for HD goodput. In the next section we discuss how we determine if a transaction that is capable of testing for HD goodput was able to *achieve* it.

G_{testable} is used to identify when a transaction cannot test for HD goodput because the response size is too small or because the session has not yet had the opportunity to grow its CWND, in which case we exclude the transaction from our goodput analysis. G_{testable} intentionally does *not* reflect the impact of actual network conditions on CWND growth. Consider how the session illustrated in Figure 4 may actually behave in production. Under poor network conditions, the CWND at the start of the third transaction may be 1 (instead of 20) because timeouts during the preceding transactions caused the CWND to be reduced. If we (incorrectly) considered W_{start} = 1, then G_{testable} = 1.4Mbps (7 packets/60ms) and we would infer that the third transaction cannot test for HD goodput. This is problematic as we only learn about network conditions when we are able to *test* for HD goodput. Worse, saying that the third transaction cannot test for HD goodput would incorrectly ignore strong evidence of poor performance. Transactions in a session being unable to test for HD goodput is in itself not a signal; it simply reflects that the session transferred small objects that could not achieve HD goodput.

To avoid this problem, we always calculate G_{testable} by setting W_{start} for each transaction assuming CWND growth under ideal circumstances. Then, the transactions identified as capable of testing HD goodput are the points in the session when, if a connection has good performance, it will (a) have a CWND capable of supporting HD goodput and (b) be delivering a large enough response to demonstrate HD goodput. We define W_{NIC} as the CWND measured when a transaction's first response byte is written to the NIC. For the first transaction W_{start} is equal to W_{NIC}. For all subsequent transactions, we define W_{start} as the maximum between W_{NIC} and the ideal CWND at the end of the previous transaction (estimated as $W_{\text{SS}}(m)$, where m is the number of round-trips in the previous transaction under ideal network conditions).[4]

[3] The Linux kernel's TCP implementation grows the CWND when the connection is CWND limited in the last round-trip and is not in loss or recovery states. Growth is determined by the number of *bytes* ACKed, not the number of ACKs received. A connection is CWND limited during slow start if it had more than half of the CWND (in bytes) in flight. After slow start a connection is CWND limited if sending was blocked on CWND. The growth for partially-utilized CWNDs is difficult to model as it depends on precisely when acknowledgements are received.

[4] $W_{\text{SS}}(m)$ provides a lower bound on the ideal W_{start} of the next transaction because it ignores any growth of the CWND during the last round-trip (footnote 3). Taking the

3.2.3 Measuring if a transaction achieved a testable goodput. The previous section described how to determine the maximum goodput that each transaction can test for under ideal network conditions, specifically focusing on maximum goodput across the transaction's RTTs. This section describes how we determine whether a transaction was able to achieve a goodput that it was capable of testing—*e.g.,* for a given transaction capable of *testing* for HD goodput, did it actually *achieve* HD goodput? Making this determination requires accounting for the impact of real network conditions (*e.g.,* packet loss and transmission time) and extrapolating behavior about goodput for one RTT to the entire transaction.

Accounting for transmission time at bottleneck links. Actual packet transfer time is composed of transmission time, propagation delay, and queueing delay. At a minimum, all responses will traverse a bottleneck link that will shape packets due to transmission delays. If we do not account for the delays, then we may say that a transaction capable of testing for HD goodput failed to achieve HD goodput in cases where goodput at the user side (which is ultimately our focus) would meet or exceed HD goodput.

Consider again transaction 3 in the session illustrated in Figure 4, but this time assume there is a 3Mbps bottleneck link between Facebook and the user. Transmission times at the bottleneck link will add ≈55ms to transaction 3's transfer time, increasing its total duration to ≈115ms.[5] With this increase in transfer time, goodput calculated will be 1.46Mbps (14 packets/115ms). Thus, if we do not account for the impact of transmission time at the bottleneck link, we will correctly infer that transaction 3 was able to test for 2.5Mbps but *incorrectly infer* that it failed to achieve it.

We do not know the achievable rate at the bottleneck link and therefore cannot correct for it directly. In the next section, we discuss a general solution for this challenge. First, we illustrate the intuition behind our approach for a transaction that should only require a single RTT to transfer (based on W_{NIC} and its response size). For such a transaction, we can approximate the rate that the response was delivered at by estimating the transaction's transmission time, which we approximate by subtracting MinRTT (an approximation of the propagation and any persistent queueing delays) from the transfer duration (*i.e.,* T_{total} − MinRTT), and then calculating the rate as $B_{\text{total}} \div (T_{\text{total}} - \text{MinRTT})$. This rate represents the speed at which the network delivered the response and captures the delay induced by bottleneck links, queuing (including any cross-traffic), shaping, and retransmissions. If this rate is greater than HD goodput, then HD goodput was achieved.

Handling transactions that require CWND growth to test goodput. We want to extend the intuition above for transactions that require multiple round-trips to complete. For large transactions, we want to look at as much of the transaction's behavior as possible, as we can learn more about path performance (*e.g.,* packet loss and jitter) by considering the whole transaction instead of just computing goodput for any individual round-trip.

A transaction's transfer time depends on its properties (*e.g.,* W_{NIC} and congestion control algorithm) and the path's performance (*e.g.,* RTT, loss, jitter, available bandwidth). At best, a transaction would be able to deliver traffic at the rate of the available bandwidth at the bottleneck link along the path. If a real transaction experiences performance degrading events, *e.g.,* exits slow start early due to CUBIC's hybrid slow start [37], experiences losses due to congestion, or has variable RTT due to cross traffic, it will deliver traffic at an overall rate lower than that available at the bottleneck link and take longer to complete.

We estimate whether a real transaction can deliver traffic at rate R by comparing it against a *model* (best-case) transaction going through a bottleneck link with available bandwidth corresponding to R. More specifically, if the real transaction's transfer time T_{total} is shorter than the model transaction's (best-case) transfer time through a bottleneck rate R, denoted $T_{\text{model}}(R)$, then it must be the case that the real transaction delivered traffic at a rate higher than R.

Our model of a best-case transaction through a bottleneck of rate R assumes best-case network conditions, removing the impact of path properties (*e.g.,* packet loss and jitter), and a simplified optimal congestion control algorithm. More precisely, we assume a congestion control algorithm that (i) doubles the CWND n round-trips (starting from W_{NIC}) until the CWND is high enough to support rate R and (ii) delivers traffic at rate R from that point on. The first assumption minimizes the time the model transaction stays in slow start,[6] and the second assumptions optimistically minimizes transmission time by considering full, perfect utilization of the bottleneck's available bandwidth. If the model transaction exits slow start after $n + 1$ round-trips, its transfer time is given by (i) the number n of round-trips in slow start, (ii) plus the transmission time of the remaining bytes, (iii) plus one round-trip for receiving the acknowledgement of the last packet:

$$T_{\text{model}}(R) = n \times \text{MinRTT} + \frac{B_{\text{total}} - \sum_{i=0}^{n} W_{\text{SS}}(i)}{R} + \text{MinRTT},$$

where we use MinRTT as the best-case RTT.

We estimate the delivery rate of a real transaction as the largest rate R for which the transaction's $T_{\text{total}} \leq T_{\text{model}}(R)$. For the case of short responses that can transfer in one RTT, then $n = 0$ and the solution is as in the previous example, *i.e.,* $R = B_{\text{total}} \div (T_{\text{total}} - \text{MinRTT})$. If a transaction is capable of *testing* for HD goodput *and* $T_{\text{total}} \leq T_{\text{model}}(\text{HD goodput})$, then we determine that the transaction achieved a delivery rate of at least HD goodput.

Validation using simulation. We validated that our goodput estimation technique accurately approximates bottleneck bandwidth under ideal network conditions using simulations in NS3.[7] We simulated transfers through 15,840 configurations, varying bottleneck bandwidth $G_{\text{bottleneck}}$ (0.5–5 Mbps), round-trip propagation delays (20–200 ms), initial CWND sizes (1–50 packets), and transfer sizes (1–500 packets). We identify configurations whose transfers can achieve the bottleneck rate by checking if $G_{\text{testable}} > G_{\text{bottleneck}}$. For these configurations, the goodput G inferred by our technique

maximum between W_{NIC} and $W_{\text{SS}}(m)$ allows us to increase the maximum testable goodput when W_{NIC} is greater than the modeled $W_{\text{SS}}(m)$.

[5] MinRTT captures (at a minimum) the transmission time for TCP headers, so we only consider the impact of transmission time at the bottleneck link on payload.

[6] We assume the model transaction increases its CWND exponentially starting from W_{NIC}. As most of Facebook's transactions are small and may finish while in slow start, assuming constant transmission rate would deterministically underestimate R (§3.2.1).

[7] NS3's TCP implementation grows the CWND by number of ACKs received (instead of bytes ACKed). We disabled delayed ACKs to force the simulator to better match CWND growth in the Linux kernel's TCP implementation (footnote 3).

never overestimates the bottleneck rate, and usually only underestimates slightly: the 99-th percentile of the distribution of the relative error $(G_{\text{bottleneck}} - G)/G_{\text{bottleneck}}$ is 0.066. Under realistic network conditions (*e.g.,* including losses and jitter), the estimated goodput could be lower (never overestimating $G_{\text{bottleneck}}$) due to the reduced available bandwidth at the bottleneck, still capturing how fast data was delivered to the destination.

3.2.4 Defining a session's HDratio. We use the approach above to compute *HDratio*, a metric that captures the ability of an HTTP session's underlying connection to sustain HD goodput. Our approach can estimate whether a transaction can test for and deliver traffic at any rate R, but in HDratio we set R to HD goodput (*i.e.,* $R = 2.5$ Mbps). We define HDratio for each HTTP session as the ratio between the number of its transactions that achieved a delivery rate of at least HD goodput and the number of its transactions that tested for HD goodput (*i.e.,* transactions with $G_{\text{testable}} \leq 2.5$ Mbps are ignored). We compute HDratio for an HTTP session instead of individual transactions to avoid overrepresenting paths that carry sessions with many transactions (Figure 3).

3.2.5 Other Considerations.

Delayed ACKs. TCP delays sending an ACK until it has two packets to ACK or until an implementation-dependent timeout (30ms+ for Linux). Delayed ACKs can significantly inflate T_{total} for small responses and lead to underestimation of the achieved goodput; we avoid this by ignoring the last data packet (and its ACK). Instead, T_{total} is the interval between the instant when the first byte of the response is written to the NIC and the instant when an ACK covering the second-to-last packet of the transaction is received by the NIC, and B_{total} is the total amount of bytes transferred minus the number of bytes in the last packet.[8]

HTTP/2 Preemption and Multiplexing. HTTP/2 preempts and multiplexes transactions based on the transactions' priorities [12]. Sending of a transaction response is preempted (paused) if a higher priority transaction is ready to send, and the HTTP/2 send window is multiplexed when transactions have equal priority. When preemption or multiplexing occurs, a transaction's T_{total} may be inflated because it includes time spent transferring other responses. To prevent inflation, we coalesce transactions together into a single larger transaction when their responses are multiplexed or preempted. In addition, we coalesce transactions when their responses are written back-to-back to enable a sequence of small responses to be considered as a single large response.[9]

Bytes in Flight. Our approach to calculating goodput assumes that no response bytes are in flight when a new transaction's first response byte is sent. To preserve this requirement, a transaction is ineligible to be used for goodput measurements if a previous transaction's response was still in flight (e.g., last byte not yet ACKed) when the first byte of its response was sent, but the conditions for coalescing were not met.

[8] If the ACK for the second-to-last packet is delayed, it will be sent when the timeout expires or when the last packet arrives at the receiver. Thus, the delay for the second-to-last ACK is no larger than the guaranteed timeout incurred for delayed last ACKs. This approach performs no worse than the naive approach, and is more accurate in the common case of the last two packets transmitted close in time.

[9] If two responses are written in series and the last byte of the first response has *not* been written to the NIC before the first byte of the second response is written to the send buffer, then there is no gap between the writes at the transport layer. We use socket and NIC timestamps to capture this level of detail.

3.3 Aggregating Measurements

Our focus is the performance of the routes between the edge of Facebook's PoPs and groups of users—not the performance of any single session. Internet routing is inherently focused on groups of endpoints—a BGP prefix represents a (logically arbitrary) aggregate of endpoints in the same destination network, and traffic engineering systems are limited to the routes available to each BGP prefix when making routing decisions [55, 68].

We define *user groups* as aggregates of users in the same destination network (autonomous system) that are likely to experience similar performance (*e.g.,* served by the same PoP, route, and in the same geographic location). Our approach to aggregating users balances the advantages of aggregating data finely (*e.g.,* the ability to see events that impact a small group of clients or that impact clients for a short period of time) with the requirement to have sufficient samples for a statistically significant result.

A user group is defined as the following:

- Facebook point of presence containing the server,
- Client BGP IP prefix (and thus inherently the client AS), and
- Client country.

We must include the BGP prefix in the grouping (instead of aggregating to the AS) because the primary and alternate routes to users in the same autonomous system can vary by BGP prefix, and thus we must determine whether performance-aware routing (§6) can improve performance on a per-prefix basis.

We include geolocation information because we find that it reduces variability relative to aggregating to the BGP prefix alone. Network address space loosely correlates with location; two user IP addresses in the same /24 are likely to be in the same geographic location [33, 35, 46]. However, a BGP prefix can contain a large block of address space and thus is more likely to include users spread widely geographically. For example, Figure 5 shows an example of a /16 prefix that serves clients in California and Hawaii and how this results in MinRTT changes over time. We experimented with deaggregating BGP prefixes (e.g., splitting a /16 into /18s or /20s) and geolocating clients into finer granularities (states and "tiles" [16]) but found that this offered minimal reductions in variability while reducing coverage when deaggregation leaves too few measurements.

We group measurements for each user group into 15 minute *time windows* to enable temporal analysis of both degradation (§5) and opportunity for performance aware routing (§6). We refer to measurements for a user group during a time window as an *aggregation*. We choose a 15 minute window to balance insights into brief events with the need to have sufficient measurements for statistically significant results. For each aggregation, we capture the 50th percentile (median, p50) of MinRTT across all sessions, denoted $MinRTT_{P50}$, and the median HDratio across sessions that had at least one transaction test the connection's ability to achieve HD goodput, denoted $HDratio_{P50}$. We aggregate MinRTT and HDratio to percentiles instead of taking the average to focus on shifts in the distribution. Taking the percentiles also avoids skew in the underlying distributions: we observe MinRTT values in the tail of the

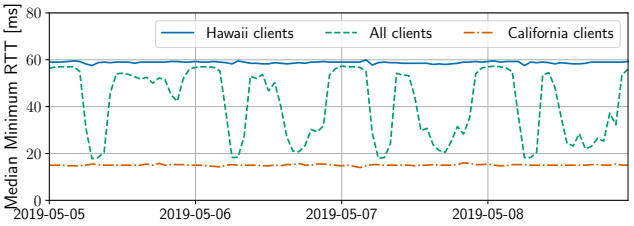


Figure 5: Example of how shifts in client population can lead to changes in MinRTT. In this example, clients in each region have stable median MinRTT, but the global median MinRTT decreases to 20ms during peak hours in California, increases to 60ms during peak hours in Hawaii, and oscillates between these two extremes during other periods.

distribution having values on the order of seconds [51], likely either due to bufferbloat [34] or last-mile / last-link problems; and the distribution of HDratio is frequently bimodal as sessions frequently have HDratio at the extremes 0.0 and 1.0.[10]

When reporting performance across aggregations, we weight results by the volume of traffic in the corresponding HTTP sessions. We focus on traffic volumes because prefixes are arbitrary units of address space whose size may not map to the underlying userbase size [27, 39] and can be subdivided arbitrarily [13].

3.4 Comparing Performance

Internet performance can vary over time due to failures, routing changes, traffic engineering, transient congestion (*e.g.,* during peak hours), changes in client population (*e.g.,* as users change networks at the end of working hours or as users in earlier time zones go to sleep). To capture these effects, we evaluate performance over time to assess whether we can identify temporal patterns of Internet performance. We characterize performance degradation to check how the performance for a user group changes over time, *e.g.,* is performance at 6am significantly better than performance at 8pm? Facebook PoPs have multiple paths to most destination prefixes they serve. For these user groups, we also compare the performance of available paths to characterize the opportunity to improve performance by choosing the best performing path, *e.g.,* when using a performance-aware traffic engineering system [55, 68].

We compute performance *degradation* over time by identifying the *baseline* performance of each user group, then comparing performance of the (BGP) preferred route in each time window against the baseline. We define the baseline MinRTT_{P50} of a user group as the 10^{th} percentile of the MinRTT_{P50} distribution of its preferred route across all time windows, and the baseline HDratio_{P50} as the 90^{th} percentile of the corresponding distribution. We consider an aggregation to be experiencing performance degradation whenever the lower bound of the confidence interval of the difference between the baseline performance and the current performance is above a configurable threshold (e.g., 5ms for MinRTT_{P50}). We compare the lower bound of the confidence interval to check for aggregations where there likely *is* degradation at the chosen threshold.

[10]We have also reproduced our analysis in §6 comparing the average HDratio across aggregations (omitted), with qualitatively similar results and findings.

We also compute the *opportunity* to improve performance over time by using alternate routes. Within an aggregation, we compare the performance of the preferred route with the performance of the best alternate route. We consider there to be an opportunity to improve performance whenever the lower bound of the confidence interval of the performance difference between the preferred and best alternate routes is above a configurable threshold (e.g., 0.05 for HDratio_{P50}). Given HDratio captures a richer view of performance, we assume networks would prioritize improving HDratio over MinRTT. When an alternate route has better MinRTT_{P50} at the specified threshold, we classify it as an opportunity for improving MinRTT_{P50} only if the HDratio_{P50} of the alternate route is statistically equal to or better than that of the preferred route.

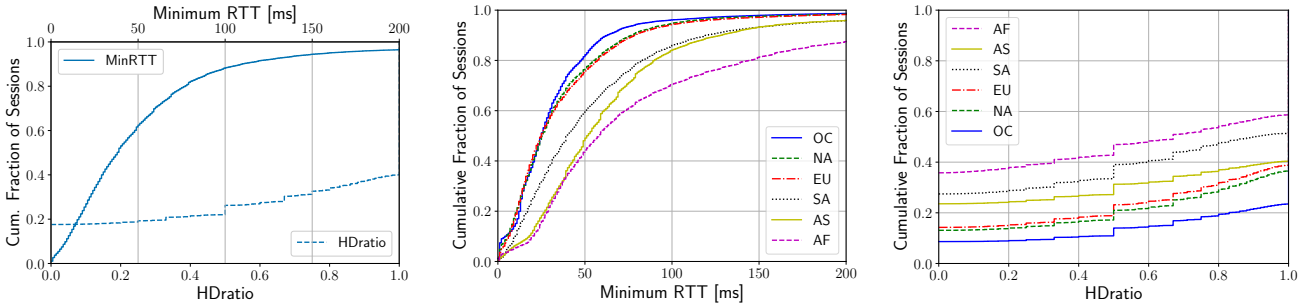
3.4.1 Controlling Statistical Significance. The statistical significance of our observations depends on the underlying performance variance and the number of samples. If clients in a user group have similar performance, then few samples are needed to obtain a good estimate of their performance; conversely, if clients in a user group have highly variable performance, then more samples are needed. Our approach allows us to not focus on a target number of samples, but instead on whether the comparison is of enough precision to support conclusions.

When comparing two aggregations (baseline vs current performance for degradation or primary vs best alternate for opportunity), we compute their performance difference and the $\alpha = 0.95$ confidence interval of the difference. Since we cannot assume normality, we compute the confidence interval for the difference of medians for MinRTT_{P50} and HDratio_{P50} using a distribution-free technique [52].[11] We only consider aggregations with at least 30 samples, and define comparisons of aggregations to be *valid* for analysis when we can calculate "tight" confidence intervals: we require the confidence intervals of the differences to be smaller than 10ms for MinRTT_{P50} and 0.1 for HDratio_{P50}. Using larger thresholds allows us to capture more traffic at a lower statistical significance, and smaller thresholds provide additional statistical significance at the cost of invalidating more time windows. We find that thresholds defining confidence intervals with half and double the chosen sizes yield qualitatively similar results (not shown).

3.4.2 Temporal Behavior Classes. After we compute degradation/opportunity over time, we try to identify temporal patterns. In particular, we identify user groups that have persistent or diurnal degradation/opportunity. We classify each user group into one of the following classes, checking the conditions for each in order:

1. The *uneventful* class includes user groups where no valid time window has degradation/opportunity. This class captures user groups where performance is stable over time (no degradation) or the preferred route is consistently better than the best alternate route (no opportunity).
2. The *persistent* class includes user groups with degradation/opportunity for at least 75% of the time windows. This class

[11]Traffic engineering systems in production need to be able to make these comparisons in near real-time (for instance, to compare the performance of routes to a network). *t-digests* (an on-line, probabilistic data structure [30]) can be used to efficiently calculate percentiles in streaming analytics frameworks and calculate confidence intervals via the cited approach.



(a) All Sessions (b) MinRTT, Per Continent (c) HDratio, Per Continent

Figure 6: Distribution of MinRTT and HDratio over all sessions and split per continent. Most sessions have low RTTs, and most sessions that test for HD goodput are able to achieve it for all transactions that can test for it.

captures user groups with frequent degradation or where the alternate route is often better than the preferred route.

3. The *diurnal* class includes user groups with degradation/opportunity for at least one fixed 15-minute time window (e.g., 11:00–11:15) in at least 5 days in our dataset. This class captures user groups where there is degradation/opportunity for part of the day over multiple days.
4. The *episodic* class includes all remaining user groups. It captures user groups with some degradation/opportunity but that do not fit into the consistent or diurnal classes.

As we need a representative view of a user group's behavior over time to classify its behavior, we ignore user groups that have traffic for less than 60% of the time windows. This can happen, for example, due to the aggregation only sporadically having client traffic (*e.g.,* business networks with very little traffic off hours) or Cartographer [56] redirecting clients to different PoPs.

The definitions above make the uneventful class restrictive (excludes user groups with *any* opportunity/degradation), while the other classes are somewhat inclusive (*e.g.,* any user group with *one* time window that experiences repeated opportunity/degradation will be classified as diurnal). Results presented in §§ 5 and 6 are robust to and findings qualitatively similar for different thresholds in the classification algorithm.

4 OVERVIEW OF GLOBAL PERFORMANCE

In this section we present a snapshot of Internet performance for users around the world, as measured from Facebook's perspective. The breadth and diversity of Facebook's users—Facebook serves billions of users from hundreds of countries every day—and the fact that Facebook is a large content provider yields an opportunity to explore and compare network performance across regions worldwide.

Users typically have low RTTs to PoPs. Figure 6(a) shows the distribution of MinRTT per session. We observe that 50% of sessions have MinRTT less than 39 milliseconds, and 80% of sessions have MinRTT of less than 78 milliseconds. Figure 6(b) depicts the MinRTT distribution by continent. The median latency in Africa is 58ms, Asia is 51ms, and South America is 40ms. The median in other continents is approximately 25 milliseconds or less.

These results indicate that most users reach Facebook over routes with low MinRTT, enabling real-time applications such as video calls. Performance tends to be better in continents with more developed infrastructure [25], both in terms of access networks and density of Facebook PoPs.

Users are typically able to achieve HD goodput. Figure 6(a) shows the distribution of HDratio across HTTP sessions. Over 82% of sessions have an HDratio greater than zero. This means that at least some of these sessions' transactions were able to achieve HD goodput, indicating that the underlying routes can support HD goodput when there is no congestion. Because our approach uses small transactions, it is inherently impacted more by jitter, particularly on connections with low propagation delay (as measured by MinRTT). For instance, if our model shows that a given transaction should complete in 2 ms and it actually completes in 4 ms (due to 2 ms jitter), then it may be tagged as non-HD-capable. Further, 60% have HDratio of 1, meaning most sessions have enough bandwidth to support HD video. We evaluated the benefit of our approach by comparing it to a simple approach that estimates the achieved goodput of a transaction as its overall goodput $B_{\text{total}} \div (T_{\text{total}})$ (but still uses our techniques from §§ 3.2.2 and 3.2.5). The simple approach underestimates which transactions reach HD goodput, yielding an underestimated median HDratio of 0.69.

Figure 6(c) shows that HDratio follows a similar per-continent trend as MinRTT, with Africa, Asia and South America standing out for having more sessions with HDratio equal to zero: 36% of African sessions, 24% of Asian sessions, and 27% of South American sessions. This result indicates a higher concentration of clients with non-HD-capable access links in these regions.

MinRTT does not directly correlate with HDratio. A transaction's ability to test for HD goodput is partially dependent on latency; sessions with higher latency require larger transfers to test for and achieve HD goodput (§3.2). However, because our approach to calculating HDratio considers whether a transaction is able to test for HD goodput based on its latency, response size, and CWND, HDratio does not necessarily decrease as latency increases. Instead, achieving HDratio at higher latencies is dependent on the loss rate, jitter, and other aspects.

Figure 7 shows how MinRTT and HDratio correlate. We group by ranges of MinRTT and show the distribution of HDratio for each range. HDratio degrades as latency increases, but the majority of sessions achieve HD goodput for some transactions even at MinRTT

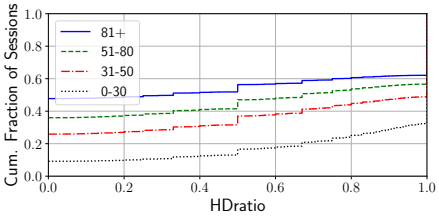
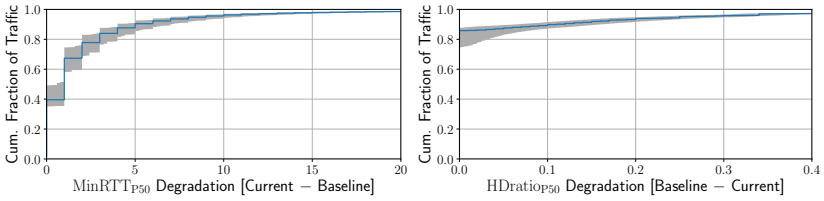


Figure 7: Relationship between MinRTT (different lines) and HDratio (*x* axis). Sessions with higher MinRTT values are often still able to achieve HD goodput.



(a) MinRTT$_{P50}$ (b) HDratio$_{P50}$

Figure 8: Degradation in MinRTT$_{P50}$ and HDratio$_{P50}$, comparing the performance of each time window with the baseline performance for the same user group. The shaded areas show the *distributions* of the lower and upper bounds of the degradation confidence interval (not the confidence interval around individual points on the CDF), and provide an indication of where the distribution is.

above 80 milliseconds, indicating that users in these groupings have connections capable of supporting greater than 2.5Mbps. Thus, the largest barrier to these clients achieving HD goodput is likely the impact of loss and traffic policing; with high latency, even a small loss rate makes maintaining a given goodput difficult [20, 31, 50].

5 HOW DOES PERFORMANCE CHANGE OVER TIME?

In this section we look at how both of our performance metrics change over time. Performance for a given destination may vary for a number of reasons, including path changes (due to routing dynamics or traffic engineering [55, 56]), congestion, or changes in the client population. We compute performance degradation (§3.4) for each user group to quantify how much user-perceived performance changes over time.

After removing any aggregation with insufficient traffic or too large of a confidence window (§3.4), the results for MinRTT$_{P50}$ include 94.8% of traffic and results for HDratio$_{P50}$ include 89.5% of traffic.

Figure 8 shows the distributions of degradation for MinRTT$_{P50}$ and HDratio$_{P50}$, comparing the difference in performance for each aggregation to the baseline for the user group, weighted by the volume of traffic of each aggregation to enable the impact of a degradation to be better understood. The vast majority of traffic sees minimal degradation over the 10 days in the study period, with only 10% of traffic experiencing 4 millisecond or worse degradation in MinRTT$_{P50}$ and 10% experiencing a 0.065 or worse degradation in HDratio$_{P50}$, both of which can be the result of minor changes in client population, or client behavior. However, in the tail we observe 1.1% of traffic experiencing degradation of at least 20 milliseconds, and 2.3% of traffic has a degradation of at least 0.4 in HDratio$_{P50}$. These changes are more significant and may indicate congestion between our PoP and the destination.

Table 1 shows degradation, computed at different thresholds, per temporal behavior class (§3.4). (The caption explains how to interpret the table.) For example, user groups responsible for 13.4% of overall traffic are classified as experiencing diurnal HDratio$_{P50}$ degradation of at least 0.05 (Entries mentioned in text are underlined and bold in table.) However, only a fraction of this traffic experienced degradation: the second number shows that 8.6% of overall traffic was delivered for these user groups during these periods of diurnal degradation.

The orange columns of the table show that most of the performance degradation is diurnal, which can be explained by degradation occurring due to congestion at peak periods. As we will discuss further in Section 6, we expect this congestion to be located at destination networks (e.g., at the last mile or peering links). We also observe that any performance degradation is usually small: the fraction of traffic experiencing degradation decreases as the threshold increases. For example, only 1.1% of traffic experience diurnal degradation of 20ms or more.[12] Finally, a significant fraction of user groups experience episodic degradation; we note, however, that the amount of traffic actually impacted is small (compare total traffic in blue vs impacted traffic in orange).

Results for individual continents follow similar trends, with Africa, Asia, and South America experiencing above-average degradation, and Oceania experiencing below-average degradation.

6 CAN PERFORMANCE BE IMPROVED VIA ROUTING CHANGES?

This section evaluates if it is possible to improve network performance for Facebook's users by changing how traffic is routed from our PoPs. This is one of the key motivations behind traffic engineering systems used by large content providers [55, 68]. Because BGP does not communicate route performance characteristics such as latency, packet loss, or the presence of congestion (all of which determine achievable goodput/HDratio), Facebook and other network operators have traditionally resorted to using heuristics to guide their routing decisions. We begin by briefly discussing these heuristics and how they correlate with performance; prior work discusses Facebook's and other network's policies in detail [55, 68]. We then compare MinRTT$_{P50}$ and HDratio$_{P50}$ across different routes in an aggregation to identify cases where performance can be improved.

6.1 Facebook's Routing Policy

A typical Facebook PoP interconnects with tens or hundreds of peers and often at least one transit provider (frequently multiple for redundancy). In most cases, the PoP serving a given user learns three or more distinct paths to the user: paths announced by one or more peers, and paths announced by two or more transit providers.

[12] The fraction of traffic experiencing diurnal and episodic degradation *can* increase as the threshold increases. Prefixes experiencing continuous or diurnal degradation at low thresholds may experience only diurnal or episodic degradation at high thresholds.

Class/ Continent	Periods of degraded performance (§5): MinRTT_{P50} (§§ 3.1, 3.3 and 3.4) +5ms		+10ms		+20ms		+50ms		HDratio_{P50} (§§ 3.2 to 3.4) −0.05		−0.1		−0.2		−0.5		Opportunity for performance-aware routing (§6): MinRTT_{P50} −5ms		−10ms		HDratio_{P50} +0.05	
Uneventful	.575		.705		.809		.929		.598		.625		.655		.742		.890		.943		.844	
AF	.344		.561		.710		.837		.541		.541		.544		.713		.570		.740		.722	
AS	.378		.518		.688		.880		.481		.487		.494		.507		.711		.828		.798	
EU	.637		.747		.813		.932		.590		.634		.688		.754		.939		.977		.857	
NA	.680		.813		.909		.984		.656		.671		.681		.817		.916		.961		.839	
OC	.899		.955		.976		.993		.662		.662		.662		.672		.901		.976		.688	
SA	.296		.454		.633		.817		.497		.501		.541		.721		.583		.676		.913	
Continuous	.008	.007	.002	.001	.000	.000	.000	.000	.019	.018	.009	.009	.008	.008	.001	.001	.013	**.012**	.006	.006	.000	.000
AF	.017	.015	.002	.001	.000	.000	.000	.000	.212	.202	.212	.199	.212	.196	.052	.051	.119	.115	.043	.041	.000	.000
AS	.006	.005	.001	.001	.000	.000	.000	.000	.035	.033	.035	.033	.035	.033	.001	.001	.049	.046	.036	.034	.000	.000
EU	.007	.006	.003	.003	.000	.000	.000	.000	.021	.020	.001	.001	.000	.000	.000	.000	.004	.004	.000	.000	.000	.000
NA	.009	.007	.000	.000	.000	.000	.000	.000	.003	.003	.002	.001	.000	.000	.000	.000	.004	.003	.001	.001	.000	.000
OC	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.004	.004	.004	.004	.000	.000
SA	.010	.007	.001	.000	.000	.000	.000	.000	.011	.011	.011	.011	.011	.011	.000	.000	.072	.069	.046	.043	.000	.000
Diurnal	.175	.060	.091	.023	.043	.008	.010	.002	**.134**	**.086**	.135	.075	.089	.043	.017	.009	.016	.007	.005	.002	.005	.001
AF	.312	.149	.183	.092	.126	.047	.069	.011	.091	.059	.091	.035	.085	.012	.083	.065	.094	.049	.069	.031	.000	.000
AS	.322	.125	.166	.041	.064	.011	.022	.003	.075	.054	.069	.051	.065	.049	.085	.054	.035	.010	.012	.004	.003	.002
EU	.149	.035	.082	.012	.045	.004	.002	.000	.135	.076	.143	.066	.059	.028	.006	.001	.005	.002	.001	.000	.009	.002
NA	.075	.026	.033	.009	.011	.003	.002	.000	.154	.108	.151	.096	.132	.062	.005	.001	.015	.008	.001	.001	.000	.000
OC	.034	.009	.018	.003	.008	.001	.002	.000	.002	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
SA	.383	.164	.174	.063	.082	.023	.018	.004	.234	.174	.234	.157	.197	.070	.020	.010	.108	.036	.062	.027	.000	.000
Episodic	.242	.007	.202	.005	.148	.003	.061	.001	.249	.002	.231	.001	.249	.002	.239	.001	.081	.001	.046	.001	.151	.001
AF	.327	.007	.255	.003	.164	.002	.094	.002	.156	.001	.156	.000	.159	.000	.153	.000	.217	.002	.148	.006	.278	.001
AS	.294	.012	.315	.012	.247	.006	.099	.002	.410	.002	.410	.002	.407	.002	.406	.001	.205	.007	.124	.003	.200	.001
EU	.207	.006	.167	.004	.142	.003	.066	.001	.253	.002	.222	.001	.253	.001	.240	.001	.052	.001	.022	.000	.134	.001
NA	.236	.004	.153	.003	.080	.001	.014	.000	.186	.002	.176	.001	.187	.002	.178	.001	.064	.001	.036	.000	.160	.001
OC	.067	.001	.027	.001	.016	.000	.005	.000	.336	.001	.338	.001	.338	.001	.328	.001	.095	.001	.020	.000	.312	.002
SA	**.312**	**.011**	.371	.015	.285	.010	.164	.004	.258	.001	.254	.001	.250	.001	.258	.004	.237	.004	.217	.003	.087	.000

Table 1: Fraction of traffic on prefixes by temporal behavior class (§3.4.2) and user geographic location for periods of degraded performance (§5) and periods with opportunity for performance-aware routing (§6) at various thresholds of degradation/improvement (§3.3). In each pair of columns, each user group is assigned a single class and a single continent. The first blue column weights the user group by its total traffic volume, normalized overall and per continent, and so the classes sum to 1 per column (across classes, without continent breakdown) and to 1 per column per continent. This column provides insight into the amount of users (weighed by their total traffic) that experienced the temporal behavior, and therefore provides insight into how widespread the events are. The orange column shows the fraction of overall (or per continent) traffic sent from those PoPs to those prefixes *during* the episodes of performance degradation or of opportunity for improvement via performance-aware rerouting. This column provides insights into the amount of traffic associated with the episodes. For example (bottom leftmost entries), 31.2% of traffic to South American clients is for user groups that experience episodic degradation of at least 5ms, and 1.1% of traffic to South American clients is sent during the episodes of degradation.

When a Facebook PoP has multiple routes to a user, it decides among them by applying the following four tiebreakers in order: (1) Prefer longest matching prefix, (2) Prefer peer routes, (3) Prefer shorter AS-paths, (4) Prefer routes via a private network interconnect (PNI).

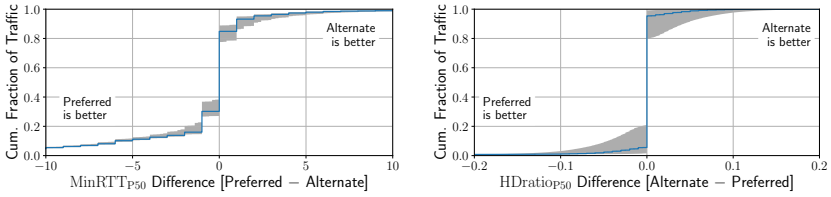
Peer routes are preferred because they are more likely to be short, direct routes into the destination network with lower latency and better performance compared to routes via transit providers [5, 26]. Further, from operational experience we know that routes via transit providers frequently lack the capacity required to deliver Facebook's traffic to a destination, resulting in congestion (§6.2.2). Paths via a private network interconnect (PNI) are preferred over routes via public exchanges (IXPs) because Facebook can monitor PNI circuit capacity and utilization to prevent congestion (§2.2.3).

6.2 Opportunity for Performance-aware Routing

In this section we quantify the opportunity for improving performance by shifting traffic from the policy-defined preferred route to an alternate route (§3.4). Approximately 47% of sampled HTTP sessions are routed via the best path to the destination (regardless of the decisions made by Facebook's egress routing controller, as described in §2.2.3). The remaining sessions are used to measure the performance of alternate paths, which by default are the two next best paths to the destination (as determined by the policy described in §6.1). For each aggregation, we compare the preferred route and the best performing alternate. We find 89.5% of the traffic has valid aggregations (at least two routes and "tight" confidence intervals) for 60% of the time windows when computing opportunity to improve MinRTT_{P50} (85.8% of traffic for HDratio_{P50}).

The solid line in Figure 9 shows the performance difference between the preferred and best alternate routes for all valid aggregations. The distributions are concentrated around $x = 0$, indicating that the preferred and best alternate routes often have similar performance. The MinRTT_{P50} of the preferred path is within 3ms of optimal (*i.e.,* whichever of *preferred* and *best alternate* performs better) for 83.9% of traffic, and the HDratio_{P50} of the preferred path is within 0.025 of optimal for 93.4% of traffic. Overall, we find few opportunities to improve performance: MinRTT_{P50} can be improved by 5ms or more for only 2.0% of traffic, and HDratio_{P50} can be improved by 0.05 or more for only 0.2% of traffic. One possible explanation for finding less opportunity to improve HDratio_{P50} compared to MinRTT_{P50} is that congestion often occurs at the destination network (*e.g.,* regional ISPs with limited infrastructure or access links); in these cases, congestion cannot be bypassed by using alternate routes as both converge at the destination network and traverse the bottleneck. MinRTT_{P50}, however, is not defined by a single bottleneck and can improve whenever a better alternate route is available. The difference distribution for MinRTT_{P50} has more density on $x < 0$ (*i.e.,* they are skewed to the left), which means that the preferred route is more likely to outperform the best alternate route than the opposite.

6.2.1 When and Where Are the Opportunities for Improvement? Table 1 breaks down opportunity for improving MinRTT_{P50} by 5ms and HDratio_{P50} by 0.05 by temporal pattern and client continent. We prioritize HDratio_{P50} when assessing connection performance



(a) MinRTT$_{P50}$ (b) HDratio$_{P50}$

Figure 9: Possible performance improvement, weighted by traffic, over 15 minute time windows. Positive values mean the alternate path is better than the primary path (lower MinRTT$_{P50}$, higher HDratio$_{P50}$). The shaded areas show the *distributions* of the lower and uppwer bounds of confidence intervals.

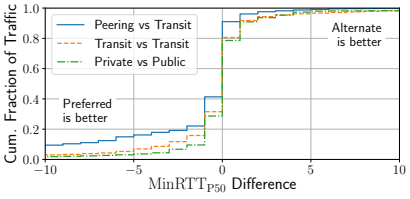


Figure 10: MinRTT$_{P50}$ difference (§§ 3.1, 3.3 and 3.4) of alternate routes by relationship type, weighted by traffic.

and exclude cases where MinRTT$_{P50}$ improves if HDratio$_{P50}$ degrades (§3.4). Results for higher improvement thresholds are qualitatively similar, but apply to smaller fractions of traffic (not shown). We report on lower thresholds as we identify few opportunities.

We find that most (1.2% of overall traffic) opportunity for improving MinRTT$_{P50}$ is for user groups classified as continuous, meaning that the preferred route usually has higher RTT than the best available route. We find few diurnal or episodic opportunities to improve MinRTT$_{P50}$ and even fewer episodic opportunities to improve HDratio$_{P50}$ using alternate routes. Opportunity to improve performance over alternate routes implies that the access network/technology is not the performance bottleneck. Opportunities to improve MinRTT$_{P50}$ and HDratio$_{P50}$ may be due to few congestion events happening in the non-shared portion of alternate paths (*i.e.,* intermediate networks). Opportunities to improve MinRTT$_{P50}$ may also arise due to temporary path changes (*e.g.,* when the normal path is unavailable or a different alternate route is available). Since the events are episodic, they suggest that congestion or path changes may have occurred due to a failure/maintenance, and not a long-standing bottleneck. As with degradation, we find more opportunity in Africa, Asia and South America, and less in Oceania.

6.2.2 Are Opportunities Practical? This section investigates implications of using a dynamic traffic engineering system to direct traffic to better performing paths when there is opportunity. Table 2 breaks down the traffic with opportunity for performance improvement (orange columns in Table 1) by the peering relationships of the preferred and alternate routes. A significant fraction of opportunity happens when the preferred and alternate routes have the same relationship (blue rows). In these cases, the alternate routes are often less preferred (*i.e.,* not chosen) due to having a longer AS-path compared to the preferred route. We also inspect how often the alternate route has more prepending than the preferred route, as this may be a signal of ingress traffic engineering (perhaps the route is better performing, but capacity constrained) [23], meaning that the alternate route should be deprioritized and thus is not a good candidate for improving performance. An additional fraction of opportunity is on traffic sent over private and public exchange peering links that shows better performance on transit (orange rows). Results are qualitatively similar for MinRTT$_{P50}$ and HDratio$_{P50}$, although transit providers account for more opportunity for improving HDratio$_{P50}$.

Relationships	Absolute	Relative	Longer	Prepended
MinRTT$_{P50}$ (§§ 3.1, 3.3 and 3.4)				
Private → Private	.0118	.489	.449	.327
Private → Transit	.0046	.191	N/A	N/A
Public → Public	.0001	.003	.003	.002
Public → Transit	.0021	.086	N/A	N/A
Transit → Transit	.0026	.108	.048	.027
Others	.0029	.122	N/A	N/A
HDratio$_{P50}$ (§§ 3.2 to 3.4)				
Private → Private	.0003	.081	.066	.023
Private → Transit	.0014	.398	N/A	N/A
Public → Public	.0000	.001	.001	.000
Public → Transit	.0003	.091	N/A	N/A
Transit → Transit	.0012	.361	.015	.043
Others	.0002	.068	N/A	N/A

Table 2: Opportunity to improve MinRTT$_{P50}$ and HDratio$_{P50}$ by relationship type of preferred and alternate routes (*private* interconnects, *public* IXPs, and *transit* providers). Blue rows correspond to opportunity over alternate routes of the same relationship type, and orange rows correspond to cases where a transit performs better than a peer. The *absolute* column shows the fraction of total traffic with opportunity, and the other columns show the fraction of opportunity in each relationship (*relative* column adds up to 1 for MinRTT$_{P50}$ and 1 for HDratio$_{P50}$). We show the fraction of opportunity where the routing decision occurred because the alternate route's AS-path was *longer*, as well as the fraction where it was *prepended more* than the preferred route.

Although alternate routes can support measurement traffic, they may lack the capacity to support *all* Facebook traffic to the destination [55]. Facebook's high traffic volumes can congest even Tier-1 networks; transit networks may lack capacity to deliver the additional traffic; and peer networks may not expect (or be willing to carry) the sudden increases in traffic. In practice, a traffic engineering system that simply shifts traffic onto the best performing alternate route may cause congestion and risk oscillations. An active traffic engineering system would need to gradually shift traffic onto the alternate route, continuously monitor its performance, and guarantee convergence to a stable state. Multiple large content providers operating such systems could interact in complex ways, presenting significant challenges in achieving fairness and stability.

6.3 Comparing Peer and Transit Performance

We now compare the performance of routes by the type of peering relationship. Our goal is to quantify how peer routes compare to

transit routes, and how much performance differs between transits. Given two relationships r_1 and r_2, we consider aggregations where the preferred route is of type r_1 and there is at least one alternate route of type r_2. As before, we compare the difference in MinRTT_{P50} and HDratio_{P50} and ignore time windows for which we cannot compute "tight" confidence intervals (10ms for MinRTT_{P50} and 0.1 for HDratio_{P50}). Whereas our analysis of opportunity considered the best-performing alternate route, here, if multiple alternate routes of the same relationship type are available, we pick the most preferred one based on Facebook's routing policy (§6.1).

Figure 10 shows the distribution of MinRTT_{P50} difference when comparing peering vs transit, and when comparing two transits. The distributions are concentrated around $x = 0$, indicating that differences are frequently small. However, some peer routes significantly outperform alternate transits, as 10% of traffic has peer routes with at least 10ms better MinRTT_{P50} than alternate transits. All distributions are also shifted to the left, particularly when comparing peering vs transit. This means that transit rarely has better MinRTT_{P50}, which is intuitive as peer routes are usually direct (*i.e.,* have an AS-path length of 1). The distribution for transit vs transit is less skewed, but the preferred transit routes (*i.e.,* with either equal or shorter length compared to the less preferred transit route) are better than alternate transits slightly more often than not, perhaps because shorter routes correlate with better performance. The private vs public line shows that some IXP peers might present an opportunity to improve performance. In all these cases, however, utilizing the alternate route in practice requires solving the challenge of avoiding congestion and oscillations (§6.2.2).

Results for HDratio_{P50} (omitted) find the difference between peering and transit is concentrated around $x = 0$ (comparable performance) and mostly symmetrical, indicating that cases where peering outperforms transit occur as often and by as much as cases where transit outperforms peering. HDratio_{P50} difference for transit vs transit are qualitatively similar to the MinRTT_{P50} difference (small differences, and slightly skewed to the left).

7 RELATED WORK

Measuring User Performance. Measuring user performance at scale and in a representative manner is difficult without a global vantage point. Prior work has therefore tackled this piecemeal. For instance, some work focused on performance and interdomain connectivity in developing regions of Africa [25, 36, 69] while others have focused specifically on mobile network performance [9, 70].

Studies that more widely characterized user performance typically still had limited measurement coverage and vantage points [3, 5, 55, 63, 71]. These limitations made it challenging for them to characterize at high granularity (in terms of time and client population) the performance that can be achieved by a global network.

Understanding Impact of Network Metrics. Researchers have been trying to understand how performance and user experience may be impacted by different factors. Modeling TCP performance based on network parameters has helped us understand how latency and loss affect performance [8, 21, 50]. Previous work studied the impact of network and server metrics on HTTP performance [10, 44, 65] and video engagement [29]. These insights helped us understand the importance of network metrics on user performance.

Performance Aware Routing. Prior work has explored opportunities for performance aware routing across multiple dimensions. Akella *et al* focused on the impact of multihoming and the potential of improving performance by choosing different upstream ISPs[6, 7]. Spring *et al* quantified the prevalence of AS-path inflation and its impact on latency [58]. A 2014 Measurement-Lab report showed diurnal performance degradation of several access networks and concluded it was influenced by ISP interconnections [3]. Ahmed *et al* characterized the performance of peering links compared to transit links and found that the former were mostly better [5]. Our analysis in §6 is most similar to that of Akella *et al*, in that we investigate potential opportunities to improve performance by changing how traffic is routed to users. However, those studies were conducted before the era of massive peering [26].

Other control systems. Large content providers have been extending their backbone network infrastructure and establishing direct peering relationships with numerous other networks [15, 17, 26]. With the change in infrastructure, these providers have built control systems [55, 68] in order to have better control over their egress traffic and bypass BGP limitations. Those systems could benefit from our measurement insights to better understand under what conditions they can potentially improve user experience by accounting for path performance in their routing decisions. Prior work has explored challenges in mapping *ingress* requests to PoPs, including metrics for quantifying and comparing each PoP's performance and routing user traffic [19, 24, 61]. Our work is complementary in that it focuses on quantifying the performance of *egress* routes from a PoP to users and identifying opportunities to improve performance through changes to traditionally simple egress routing policies.

8 CONCLUSION

This paper provides a large-scale study of user traffic and performance from the vantage point of Facebook, a large content provider with over 2 billion users worldwide. We first dissected the properties of Internet traffic for large content providers and the challenges associated with measuring network performance passively when object sizes are small. We discussed how we collect server-side measurements, including HDratio, our novel approach to characterizing achievable client goodput. Equipped with a robust measurement methodology and dataset, we characterized Facebook traffic. Most users are able to achieve good performance, although there are regional variances, particularly in Asia, Africa, and South America. We found limited opportunity to improve performance by incorporating performance measurements into routing decisions; default routing decisions taken by Facebook are largely optimal.

Acknowledgements. We thank Facebook colleagues for insights and help, including Xintong Hu, Michael Shao, Alan Frindell, Aman Sharma, Ashley Hatch, James Quinn, Petr Lapukhov, Hyojeong Kim, Varghese Vaidhyan, Niky Riga, Luca Niccolini, Udip Pant, James Zeng, and Omar Baldonado. We also thank Anubhavnidhi Abhashkumar, Kevin Yang, Ramesh Govidan, our shepherd Rocky K. C. Chang and the IMC reviewers. Brandon Schlinker's research was partly funded by the Facebook Graduate Fellowship. Italo Cunha was partly funded by CNPq, CAPES, and a Facebook Research Award. Ethan Katz-Bassett was partly funded by Facebook Research Awards and partly by NSF grants CNS-1836872 and CNS-1835253.

REFERENCES

[1] The M-Lab NDT Data Set. https://measurementlab.net/tools/ndt

[2] Recommendation G. 114, One-way transmission time. Series G: Transmission Systems and Media, Digital Systems and Networks, Telecommunication Standardization Sector of ITU.

[3] ISP Interconnection and its Impact on Consumer Internet Performance. M-Lab.

[4] Google Video Quality Report. https://www.google.com/get/videoqualityreport/#methodology

[5] A. Ahmed, Z. Shafiq, H. Bedi, and A. Khakpour. Peering vs. Transit: Performance Comparison of Peering and Transit Interconnections. In *ICNP*, 2017.

[6] A. Akella, B. Maggs, S. Seshan, and A. Shaikh. 2008. On the Performance Benefits of Multihoming Route Control. *IEEE/ACM Transactions on Networking* 16, 1 (2008), 91–104.

[7] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A Measurement-Based Analysis of Multihoming. In *ACM SIGCOMM*, 2003.

[8] E. Altman, K. Avrachenkov, and C. Barakat. A Stochastic Model of TCP/IP with Stationary Random Losses. In *ACM SIGCOMM*, 2000.

[9] D. Baltrunas, A. Elmokashfi, and A. Kvalbein. Measuring the reliability of mobile broadband networks. In *ACM IMC*, 2014.

[10] P. Barford and M. Crovella. 2001. Critical Path Analysis of TCP Transactions. *IEEE/ACM Trans. Netw.* 9, 3 (June 2001), 238–248. https://doi.org/10.1109/90.929848

[11] M. Belshe. More bandwidth doesn't matter (much). http://docs.google.com/a/chromium.org/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2

[12] M. Belshe, R. Peon, and M. Thomson. 2015. *Hypertext Transfer Protocol Version 2 (HTTP/2),*. RFC 7540. RFC Editor. http://www.rfc-editor.org/rfc/rfc7540.txt http://www.rfc-editor.org/rfc/rfc7540.txt.

[13] R. Beverly, A. Berger, and G. Xie. Primitives for Active Internet Topology Mapping: Toward High-Frequency Characterization. In *ACM IMC*, 2010.

[14] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. 2001. *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations,*. RFC 3135. RFC Editor.

[15] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig. 2018. Open Connect Everywhere: A Glimpse at the Internet Ecosystem Through the Lens of the Netflix CDN. *SIGCOMM Comput. Commun. Rev.* 48, 1 (2018), 28–34.

[16] R. Brundritt, S. Cai, and C. French. Bing Maps Tile System. https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system.

[17] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, and R. Govindan. Mapping the Expansion of Google's Serving Infrastructure. In *ACM IMC*, 2013.

[18] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye. Analyzing the Performance of an Anycast CDN. In *ACM IMC*, 2015.

[19] M. Calder, M. Schröder, R. S. Ryan Gao, J. Padhye, R. Mahajan, G. Ananthanarayanan, and E. Katz-Bassett. Odin: Microsoft's Scalable Fault-Tolerant CDN Measurement System. In *USENIX NSDI*, 2018.

[20] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. 2016. BBR: Congestion-Based Congestion Control. *Queue* 14, 5 (2016), 50:20–50:53.

[21] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP Latency. In *IEEE INFOCOM*, 2000.

[22] R. Carlson. Developing the Web100 based network diagnostic tool (NDT). In *Passive and Active Measurement (PAM)*, 2003.

[23] R. K. C. Chang and M. Lo. 2005. Inbound Traffic Engineering for Multihomed ASs Using AS Path Prepending. *IEEE Network* 19, 2 (March 2005), 18–25.

[24] F. Chen, R. K. Sitaraman, and M. Torres. End-User Mapping: Next Generation Request Routing for Content Delivery. In *ACM SIGCOMM*, 2015.

[25] M. Chetty, S. Sundaresan, S. Muckaden, N. Feamster, and E. Calandro. Measuring broadband performance in South Africa. In *Proceedings of the 4th Annual Symposium on Computing for Development*, 2013.

[26] Y.-C. Chiu, B. Schlinker, A. B. Radhakrishnan, E. Katz-Bassett, and R. Govindan. Are We One Hop Away from a Better Internet?. In *ACM IMC*, 2015.

[27] A. Dainotti, K. Benson, A. King, B. Huffaker, E. Glatz, X. Dimitropoulos, P. Richter, A. Finamore, and A. C. Snoeren. 2016. Lost in Space: Improving Inference of IPv4 Address Space Utilization. *IEEE Journal on Selected Areas in Communications* 34, 6 (2016), 1862–1876.

[28] A. Dhamdhere, D. D. Clark, A. Gamero-Garrido, M. Luckie, R. K. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. C. Snoeren, and K. Claffy. Inferring Persistent Interdomain Congestion. In *ACM SIGCOMM*, 2018.

[29] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the Impact of Video Quality on User Engagement. In *ACM SIGCOMM*, 2011.

[30] T. Dunning and O. Ertl. 2019. Computing Extremely Accurate Quantiles Using t-Digests. *arXiv preprint, arXiv:1902.04023* (2019).

[31] T. Flach, P. Papageorge, A. Terzis, L. Pedrosa, Y. Cheng, T. Karim, E. Katz-Bassett, and R. Govindan. An Internet-Wide Analysis of Traffic Policing. In *ACM SIGCOMM*, 2016.

[32] C. Fraleigh, F. Tobagi, and C. Diot. Provisioning IP Backbone Networks to Support Latency Sensitive Traffic. In *IEEE INFOCOM*, 2003.

[33] M. J. Freedman, M. Vutukuru, N. Feamster, and H. Balakrishnan. Geographic Locality of IP Prefixes. In *ACM IMC*, 2005.

[34] J. Gettys. 2011. Bufferbloat: Dark buffers in the internet. *IEEE Internet Computing* 3 (2011), 96.

[35] M. Gharaibeh, H. Zhang, C. Papadopoulos, and J. Heidemann. Assessing Co-locality of IP Blocks. In *IEEE INFOCOM Workshops*, 2016.

[36] A. Gupta, M. Calder, N. Feamster, M. Chetty, E. Calandro, and E. Katz-Bassett. Peering at the internet's frontier: A first look at isp interconnectivity in africa. In *International Conference on Passive and Active Network Measurement (PAM)*, 2014.

[37] S. Ha and I. Rhee. Hybrid Slow Start for High-Bandwidth and Long-Distance Networks. In *Proc. PFLDnet*, 2008.

[38] S. Ha, I. Rhee, and L. Xu. 2008. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Oper. Syst. Rev.* 42, 5 (July 2008), 64–74.

[39] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister. Census and Survey of the Visible Internet. In *ACM IMC*, 2008.

[40] T. Henderson. Latency and User Behaviour on a Multiplayer Game Server. In *Intl. Workshop on Networked Group Communication*, 2001.

[41] C. Huang, D. A. Maltz, J. Li, and A. Greenberg. Public DNS system and global traffic management. In *IEEE INFOCOM*, 2011.

[42] J. Iyengar and M. Thomson. 2019. *QUIC: A UDP-Based Multiplexed and Secure Transport,*. Internet-Draft draft-ietf-quic-transport-23. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-23.txt.

[43] C. Kaufmann. BGP and Traffic Engineering with Akamai. MENOG 14. http://www.menog.org/presentations/menog-14/282-20140331_MENOG_BGP_and_Traffic_Engineering_with_Akamai.pdf

[44] B. Krishnamurthy and C. E. Wills. Analyzing Factors That Influence End-to-end Web Performance. In *Proc. of the 9th International World Wide Web Conference on Computer Networks*, 2000.

[45] N. Kuhn, G. Fairhurst, J. Border, and S. Emile. 2019. *QUIC for SATCOM,*. Internet-Draft draft-kuhn-quic-4-sat-00. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-kuhn-quic-4-sat-00.txt.

[46] Y. Lee and N. Spring. Identifying and Aggregating Homogeneous IPv4 /24 Blocks with Hobbit. In *ACM IMC*, 2016.

[47] J. Liddle. 2008. Amazon found every 100ms of latency cost them 1% in sales. *The GigaSpaces* 27 (2008).

[48] P. Maynard-Koran. Fixing the Internet for Real Time Applications: Part II. https://technology.riotgames.com/news/fixing-internet-real-time-applications-part-ii

[49] E. Nygren, R. K. Sitaraman, and J. Sun. 2010. The Akamai Network: A Platform for High-performance Internet Applications. *SIGOPS Oper. Syst. Rev.* 44, 3 (2010), 2–19.

[50] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. 1998. Modeling TCP Throughput: A Simple Model and Its Empirical Validation. *SIGCOMM Comput. Commun. Rev.* (1998), 303–314.

[51] R. Padmanabhan, P. Owen, A. Schulman, and N. Spring. Timeouts: Beware Surprisingly High Delay. In *ACM IMC*, 2015.

[52] R. M. Price and D. G. Bonett. 2002. Distribution-Free Confidence Intervals for Difference and Ratio of Medians. *Journal of Statistical Computation and Simulation* 72, 2 (2002), 119–124.

[53] L. Qiu, Y. Zhang, and S. Keshav. 2001. Understanding the performance of many TCP flows. *Computer Networks* 37, 3-4 (2001), 277–306.

[54] Sandvine. Sandvine Global Internet Phenomena Report: 1H 2014.

[55] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H. V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, and H. Zeng. Engineering Egress with Edge Fabric. In *ACM SIGCOMM*, 2017.

[56] P. Shuff. Building A Billion User Load Balancer. In *USENIX SREcon*, 2015.

[57] D. Sommermann and A. Frindell. Introducing Proxygen, Facebook's C++ HTTP framework. https://engineering.fb.com/production-engineering/introducing-proxygen-facebook-s-c-http-framework/.

[58] N. Spring, R. Mahajan, and T. Anderson. The Causes of Path Inflation. In *ACM SIGCOMM*, 2003.

[59] R. A. Steenbergen. 2009. A practical guide to (correctly) troubleshooting with traceroute. *North American Network Operators Group* (2009), 1–49.

[60] S. D. Strowes. 2013. Passively Measuring TCP Round-trip Times. *Commun. ACM* 56, 10 (Oct. 2013), 57–64.

[61] P. Sun, L. Vanbever, and J. Rexford. Scalable Programmable Inbound Traffic Engineering. In *ACM SOSR*, 2015.

[62] S. Sundaresan, M. Allman, A. Dhamdhere, and K. Claffy. TCP Congestion Signatures. In *ACM IMC*, 2017.

[63] S. Sundaresan, W. De Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. 2011. Broadband Internet Performance: A View From the Gateway. *ACM SIGCOMM Computer Communication Review* 41, 4 (2011), 134–145.

[64] S. Sundaresan, X. Deng, Y. Feng, D. Lee, and A. Dhamdhere. Challenges in Inferring Internet Congestion Using Throughput Measurements. In *ACM IMC*, 2017.

[65] S. Sundaresan, N. Feamster, R. Teixeira, and N. Magharei. Measuring and Mitigating Web Performance Bottlenecks in Broadband Access Networks. In *ACM IMC*, 2013.

[66] L. Thomas, E. Dubois, N. Kuhn, and E. Lochin. 2019. Google QUIC performance over a public SATCOM access. *International Journal of Satellite Communications and Networking* (2019).
[67] X. Xu, Y. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan. Investigating transparent web proxies in cellular networks. In *International Conference on Passive and Active Network Measurement (PAM)*, 2015.
[68] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, et al. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *ACM SIGCOMM*, 2017.
[69] Y. Zaki, J. Chen, T. Pötsch, T. Ahmad, and L. Subramanian. Dissecting Web Latency in Ghana. In *ACM IMC*, 2014.
[70] K. Zarifis, T. Flach, S. Nori, D. R. Choffnes, R. Govindan, E. Katz-Bassett, Z. M. Mao, and M. Welsh. Diagnosing Path Inflation of Mobile Client Traffic. In *International Conference on Passive and Active Network Measurement (PAM)*, 2014.
[71] Z. Zhang, M. Zhang, A. G. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian. Optimizing Cost and Performance in Online Service Provider Networks. In *USENIX NSDI*, 2010.