A Large-Scale Analysis of Deployed Traffic Differentiation Practices

Fangfan Li
Northeastern University

Arian Akhavan Niaki
University of Massachusetts Amherst

David Choffnes
Northeastern University

Phillipa Gill
University of Massachusetts Amherst

Alan Mislove
Northeastern University

ABSTRACT

Net neutrality has been the subject of considerable public debate over the past decade. Despite the potential impact on content providers and users, there is currently a lack of tools or data for stakeholders to independently audit the net neutrality policies of network providers. In this work, we address this issue by conducting a one-year study of content-based traffic differentiation policies deployed in operational networks, using results from 1,045,413 crowdsourced measurements conducted by 126,249 users across 2,735 ISPs in 183 countries/regions. We develop and evaluate a methodology that combines individual per-device measurements to form high-confidence, statistically significant inferences of differentiation practices, including fixed-rate bandwidth limits (i.e., throttling) and delayed throttling practices. Using this approach, we identify differentiation in both cellular and WiFi networks, comprising 30 ISPs in 7 countries. We also investigate the impact of throttling practices on video streaming resolution for several popular video streaming providers.

CCS CONCEPTS

• **Networks** → **Network measurement;**

KEYWORDS

Network Neutrality, Traffic Differentiation

ACM Reference Format:
Fangfan Li, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. 2019. A Large-Scale Analysis of Deployed Traffic Differentiation Practices. In *SIGCOMM '19: 2019 Conference of the ACM Special Interest Group on Data Communication, August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3341302.3342092

1 INTRODUCTION

Net neutrality, or the notion that Internet service providers (ISPs) should give all network traffic equal service[1], has driven active discussions, laws [2], and policies [12]. However, to date there have been few empirical studies of ISPs' traffic management policies that violate net neutrality principles, or their impact on stakeholders such as consumers, content providers, regulators, and legislators. In this work, we fill this gap via a large-scale study of a common form of net neutrality violations: content-based traffic differentiation that limits throughput for specific applications.

A large-scale study of net neutrality violations and their implications is long overdue, given that the most recent large-scale audits of net neutrality came a decade ago and focused on either backbone networks [27] or a single protocol (BitTorrent) [11]. In the intervening decade, the Internet has evolved in two key ways that require a new approach to auditing. First, today's dominant source of Internet traffic is video streaming from content providers, not BitTorrent. Second, users increasingly access the Internet from their mobile devices, often with a spectrum-constrained cellular connection. There is a need to conduct a study of net neutrality violations that takes these changes into account.

We address this need using 1,045,413 measurements conducted by 126,249 users of our Wehe app, across 2,735 ISPs in 183 countries/regions. From this set of raw measurements, we identify 144 ISPs with sufficient tests to confidently identify differentiation. Wehe builds on prior work for detecting traffic differentiation over mobile networks [16], however, while prior work focused on detecting differentiation on a per-device basis, we leverage our large-scale crowd-sourced data to develop more robust differentiation detection techniques. We then apply these techniques to conduct the largest-scale study of content-based differentiation practices to date.

The main contributions of this paper are the methods to detect throttling using data from a large user base, analysis of this data, and findings related to detecting fixed-rate throttling and their impact on affected apps . Beyond technical contributions, our findings have been used by a European national telecom regulator, the US FTC and FCC, US Senators, and numerous US state legislators. To complement this study and to help consumers and regulators make more informed decisions, we maintain a public website with updated analysis and data [6]. This website also contains an extended version of this paper with appendices that provide additional details of observed throttling. We now summarize our technical contributions.

Gathering a large dataset of content-based differentiation practices (§3) We perform the largest data collection of content-based differentiation practices, comprising more than 1,000,000 tests, which we continue to maintain on an ongoing basis. We adapted prior work [16] to enable such data collection at scale.

Method for reliably detecting fixed-rate throttling from crowdsourced measurements (§4) Individual crowdsourced tests are subject to confounding factors such as transient periods of poor network performance. To address this, we develop a method that reliably identifies fixed-rate throttling by leveraging tests from multiple users in the same ISP. We combine Kolmogorov–Smirnov tests,

[1] With a notable exception being *reasonable network management.*

SIGCOMM '19, August 19–23, 2019, Beijing, China

ACM ISBN 978-1-4503-5956-6/19/08…$15.00
https://doi.org/10.1145/3341302.3342092

kernel density estimators, and change point detection to identify cases of fixed-rate throttling and delayed throttling. We evaluated the methodology (§5) with controlled lab experiments from the 4 largest US cellular ISPs and found the results of using our methodology on crowdsourced data are consistent with lab experiments.

Characterizing differentiation affecting Wehe tests (§6) We conduct a multi-dimensional study of deployed differentiation policies measured by Wehe. We find different network providers using different rate limits (e.g., 1.5 Mbps and 4 Mbps) and targeting a different set of apps (e.g., YouTube vs. Netflix). We also find throttling practices that are poorly disclosed, falsely denied (by one ISP), and that change during the course of our study. Importantly, selective throttling policies potentially give advantages to certain content providers but not others, with implications for fair competition among content providers in throttled networks.

Characterizing video streaming implications of throttling (§7) We study how throttling in the US impacts video streaming resolution. We study the video resolutions selected by popular video streaming apps that are affected by throttling, and find examples where throttling limits video quality. We also find many cases where video players self-limit video resolution by default, in some cases selecting a lower resolution than throttling allows. Finally, we observe that streaming sessions experience retransmission rates up to 23%, leading to significant wasted network bandwidth that can be addressed through more efficient throttling implementations.

2 RELATED WORK

Traffic differentiation detection Traffic differentiation has been the target of study for over a decade. Originally, BitTorrent was studied by the Glasnost project [11] which manually crafted measurements to simulate BitTorrent and BitTorrent-like packet exchanges, followed by comparing the throughput distributions of exchanges with and without BitTorrent payloads. NetPolice [27] takes a different approach: detecting differentiation in backbone ISPs by analyzing packet loss behavior of several protocols (HTTP, BitTorrent, SMTP, *etc.*). Bonafide [10] is designed to detect differentiation and traffic shaping in the mobile ecosystem, but still relies on manually crafted files to specify protocols to test, supporting six application protocols. DiffProbe [17] focuses on Skype and Vonage, and detects differentiation by comparing latency and packet loss between exposed and control traffic. The Packsen [26] framework uses several statistical methods for detecting differentiation and inferring shaper details. NANO [24] uses passive measurement from users to infer the existence of traffic differentiation.

A limitation of prior work is that they did not generalize beyond a few tested applications, often used simulated traffic instead of traffic generated by real applications, and did not work from mobile devices. However, recent work [16, 20] showed that deployed differentiation policies often target specific applications based on keyword-based deep packet inspection and thus are often not triggered by simulated traffic. Chkdiff [22, 23] and Molavi Kakhki et al. [16] use application-generated traffic, but are not evaluated at scale. As we discuss below, we made substantial changes to the measurement and detection methodology to address the limitations of these approaches.

Identifying rate limiting Recent projects focus on identifying rate limiting of Internet traffic via shaping and policing. The ShaperProbe [18] project detects traffic shaping using end-to-end active probing with synthetic traffic, and it identified suspected shaping in multiple ISPs; however, it is not deployable on mobile devices and does not identify specific applications affected by shaping. Flach et al. [13] quantify traffic policing for YouTube and its impact on video-quality metrics, but this analysis does not generalize to other video providers and requires access to a content provider's servers. Our approach identifies rate limiting for multiple applications without requiring access to content providers' servers.

3 DATA COLLECTION

We now describe the data collected by the Wehe apps (available from the Google Play and iOS App Stores), which detect content-based differentiation between the device and a server under our control. Wehe is available to download from the Google Play and iOS App Stores.

3.1 Methodology

Record and replay To test for differentiation, Wehe uses the "record and replay" technique introduced by Molavi Kakhki et al. [16]. We first record the network traffic generated by an application (e.g., streaming a video using the YouTube app), and include this traffic trace in the app. When a user runs a test, Wehe then replays this traffic between the device and an Wehe server. *We emphasize that our tests* do not *contact content providers' servers.* Thus, all network traffic exchanged between the Wehe app and server are identical to what was recorded, with the exception of different IP addresses.

Wehe runs a series of back-to-back replay pairs. In each back-to-back pair, the *original* replay contains the same payloads as recorded (e.g., YouTube traffic). This exposes the original payload to network devices such as those that use deep packet inspection (DPI). The other replay in the back-to-back pair is the *control* replay, which contains the same traffic patterns (packet sizes, timings) but the original payload is obscured to evade detection by DPI devices that often rely on keyword matching in their classification [20, 21]. For the control replay, Wehe inverts the original payload bits, a technique that our prior work [21] found to evade DPI detection. Note that we do not use random bytes because they were found to trigger differentiation in ways that inverted bits do not [21].

Apps tested by Wehe For this study, Wehe uses traces recorded from YouTube, Netflix, Amazon Prime Video, NBC Sports, Vimeo, Spotify, and Skype. We selected the first five apps because video streaming is a common target of traffic differentiation [15, 20]. We include Spotify because some cellular plans indicate rate limits on streaming audio, and Skype because a telephony app may compete with cellular providers' voice services. The traces in Wehe consist of video streaming from the video apps, music streaming on Spotify, and a video call on Skype. Note that the traces are recorded by the Wehe team, and contain no information about the users running the tests. We use the following symbols to represent each app test: for YouTube, for Netflix, for Amazon Prime Video, for NBCSports, for Skype, for Spotify and for Vimeo.

When running Wehe, users can select which apps to test, and a test consists of up to two replay pairs. The Skype test uses UDP, while the others use TCP. Among TCP tests, NBCSports and

Spotify use HTTP, and the others use HTTPS. Thus our approach supports both plaintext and encrypted flows. For the tests that use HTTPS, we simply replay the exact same encrypted bytes over TCP connections between the Wehe app and a Wehe server. Note that since Wehe simply replays the trace as it was recorded, Wehe does not incorporate any dynamic behavior (e.g., adaptive bitrate streaming) that the recorded app might incorporate.

We support UDP traffic in our tests, but do not currently use QUIC traces. An open important research challenge is how to emulate QUIC congestion control, given that we cannot trivially distinguish new payload bytes from retransmissions due to header encryption.

Detecting differentiation for each test After replaying traces for an app, Wehe checks for differentiation and displays the result to the user. Wehe uses a Kolmogorov–Smirnov (KS) test [14] to compare the throughput distributions of the original and the control replays of a given application trace. Wehe samples throughput using fixed time intervals, based on the recorded trace duration: if the replay takes t seconds when recorded, each interval is $t/100$ seconds. Because our record and replay approach sends data no faster than it was recorded, we are guaranteed to have at least 100 samples for each test. However, if the test occurs in an environment where there is a bandwidth bottleneck, the replay can take more than t seconds. If so, we continue to sample at the same rate after t seconds, and thus would record more than 100 samples. Similar to Net Police [27], Wehe conducts Jackknife non-parametric resampling to test the validity of the KS statistic. Wehe indicates to the user that there is differentiation only if *both* the KS test is statistically significant (i.e., p-value less than 0.05, and the resampled KS tests lead to the same result 95% of the time) and the difference in average throughputs is significant (i.e., at least a 10% difference in average throughput) [16].

3.2 Implementation

Prior work detected differentiation using packet captures recorded at the replay server [16], assuming that packets received at the server (e.g., TCP ACK packets) came directly from the client. However, we found empirically that this is not the case, largely due to transparent TCP proxies that split the end-to-end connection into two TCP connections. In this case, the server cannot observe rate limits imposed only on the client–proxy connection. To address this, Wehe records traces both from the server side and from the client via periodic throughput measurements collected at the application layer (obtaining raw packet traces would require users to root their phones, which we wish to avoid). We use both traces to identify differentiation and the direction that is affected.

Prior work found that three back-to-back tests yielded low false positive and negative rates for differentiation detection [16]. However, anecdotal reports from Wehe users indicated that the time required to run these tests (16 minutes to test all apps) was a limiting factor in using Wehe. To mitigate this issue, Wehe first analyzes the result of one pair of back-to-back tests for an app. If there is no differentiation detected, then Wehe does not run additional tests for the app. If there is differentiation detected, Wehe runs an additional pair of back-to-back tests and reports differentiation to the user only if it is detected in both tests. The use of only one or two tests might cause higher error rates in results reported to individual app users. In §4 we analyze data from *all* tests of the same app in the same ISP across our user base to gain additional statistical confidence in our results.

3.3 Confounding factors and limitations

Wehe accounts for the following confounding factors when reporting results to users. First, bandwidth volatility (e.g., due to poor signal strength, cross traffic, etc.) could cause Wehe to incorrectly identify differentiation. To reduce the impact of this, Wehe performs multiple back-to-back tests and reports differentiation to users only when at least two pairs of tests indicate differentiation. This conservative approach may result in false negatives, where Wehe does not report differentiation to the user. In the next section, we discuss how we aggregate data across our user base to mitigate false negatives and positives due to volatility.

Second, the network may retain history such that one replay test impacts the treatment of the next replay. We instituted random ordering of original and bit-inverted replays, and found no evidence of history affecting our results.

Third, Wehe is subject to the same limitations prior work [16]: it cannot detect differentiation based on IP addresses, peering arrangements, interconnection congestion, traffic volume, or other factors independent of IP payloads. Detecting differentiation based on IP addresses, peering arrangements, and interconnection congestion would seem to require access to content servers (and/or their IPs)—Wehe alone cannot detect such cases because the paths our measurements follow are potentially different than the ones between clients and content servers.

Though outside the scope of this work, Wehe can be augmented to detect differentiation based on traffic volumes. Specifically, our tests preserve the recorded application's content stream in terms of packet timings and packet sizes, and could trigger differentiation based on those properties. However, both the inverted and original payloads could trigger the same behavior, so we would need to add a second control test (that does not look like any app's traffic volumes) to identify differentiation. Similarly, Wehe could incorporate tests using the real apps under test, in addition to our controlled ones using Wehe, to detect differentiation based on factors other than payload contents. We consider such approaches to be interesting areas for future work.

Last, there is no known API to determine a user's data plan or any differentiation policies on the plan, so we cannot compare Wehe findings with stated policies.

3.4 Ethics

Our work involves human subjects, and we took care to follow community best practices when conducting our work. Wehe collects anonymized data from user devices as part of an IRB-approved study. First, as described below, we collect only data that we deemed necessary to characterize differentiation and assess confounding factors. Second, when Wehe is opened by the user for the first time—and before any data is collected—users undergo informed consent via an IRB-approved consent form that specifies the data collected and how it is used. Once users consent, they can initiate tests; if the user does not consent, the app closes immediately. Third, data collection occurs only when users initiate tests, and users can opt out of data collection (and request deletion of data) at any time. Our data-collection and management process has been deemed GDPR-compliant.

3.5 Dataset

The data generated by Wehe tests includes throughput samples, as well as the following for each back-to-back test: (1) the server timestamp at the beginning of the test, (2) the first three octets (/24) of the client's IP address, (3) the client's mobile carrier as reported by the operating system, (4) the client's operating system and phone model, (5) the network connection type (WiFi or cellular), and (6) the coarse-grained GPS location (collected with user permission). We describe the reason for collecting each of these items below.

The timestamp allows us to identify trends over time. The carrier name allows us to identify the cellular provider for tests on cellular networks. The client's anonymized IP address information and network type allow us to identify the ISP being tested for WiFi connections[2], and to identify whether there are subnet-level differences in detected differentiation.

The coarse-grained GPS location (10 km precision) allows us to identify regional differences in ISPs' policies (e.g., in response to state-level net neutrality regulations in the US). The Wehe app first requests the geolocation of the user via the operating system's precise GPS location feature, the Wehe server then geo-codes the geolocation (i.e., looking up the city/state/country) and stores only the truncated geolocation (i.e., with 10 km precision). Users can choose not to share their GPS locations without limiting app functionality. In 15% of tests, the users opted out of location sharing.

The OS and phone model allow us to distinguish whether ISPs discriminate against these factors, or to what extent OSes and phone models might bias the results.

Summary of dataset We summarize our dataset in Table 1. Between Jan. 18, 2018 and Jan. 24, 2019, 59,326 iOS users and 66,923 Android users installed Wehe and ran at least one test.

In total, Wehe conducted 1,045,413 tests. We plot the distribution of tests over time in Figure 2 (note the log scale on the y-axis). We observe a peak of 77,000 tests on January 19, 2018, when a news article raised awareness of the app [3]. There were three other press events that raised awareness of the app; we still observe several hundred tests per day. Wehe users come from at least 183 countries based on geolocation.

Like any crowdsourced dataset, ours is subject to several biases that may impact the generality of our findings. We cannot control when, where, or why users run our tests, and thus we do not have uniform or complete coverage of any ISP or app tested. Figure 1 shows the distribution of test locations, where the intensity of the color for each country reflects the number of tests completed in the country. More than 60% of our tests come from the US, most likely due to the recent changes in net neutrality rules combined with US-centric press articles. The phone models used in our tests skew toward higher-end devices, Table 2 shows the top phone models and OSes for users in the Wehe dataset. A large fraction of our US tests come from large cellular providers, meaning lower-cost providers (e.g., MVNOs) are under-represented.

Despite these biases, our analysis covers 2,735 ISPs[3] in 183 countries, and identifies differentiation in 30 ISPs in 7 countries. We believe this to be the largest study of content-based differentiation practices.

[2] Using the "OrgName" field from whois queries to regional Internet registries.

[3] We noticed that some ISPs used multiple "OrgNames" (e.g., Bouygues and BouyguesTelecom); thus, some ISPs may be counted multiple times.

Replay	Users (%)	Cellular Tests	WiFi Tests
YouTube	106,813 (85%)	97,009	149,850
Netflix	83,369 (66%)	66,320	112,473
Amazon	77,212 (61%)	61,851	102,529
Spotify	65,644 (52%)	43,306	90,963
Skype	60,658 (48%)	37,589	72,250
Vimeo	49,701 (39%)	33,538	67,333
NBC Sports	49,605 (39%)	38,701	71,701
Total	126,249	378,314	667,099

Table 1: Overview of Wehe data analyzed in this paper.

	iOS		Android	
Users	59,326		66,923	
Top five OS versions	iOS 11.2.2	15%	Android 7.0	17%
	iOS 11.2.5	7%	Android 8.0.0	9%
	iOS 12.1	5%	Android 8.1.0	8%
	iOS 11.4.1	4%	Android 7.1.1	5%
	iOS 11.2.6	3%	Android 6.0.1	4%
Top five phone models	iPhone X	19%	Pixel 2 XL	2.2%
	iPhone 7	14%	Samsung S8	1.9%
	iPhone 6s	12%	Pixel XL	1.8%
	iPhone 7 Plus	11%	Samsung S8+	1.8%
	iPhone 6	7%	Pixel	1.7%

Table 2: Summary of Wehe users' phone models. There is a bias toward newer phones and OSes, with devices capable of displaying content in HD.

Figure 1: Number of tests per country (log scale). Note that 15% of our tests do not have GPS data (e.g., if the user did not provide permission to collect GPS locations), and we excluded them from any geolocation-based analysis.

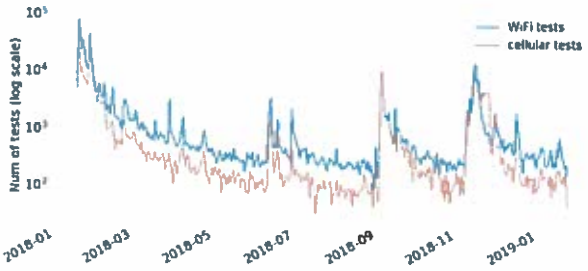

Figure 2: Number of Wehe tests per day (log scale).

4 DETECTING DIFFERENTIATION

We now describe our methodology for identifying and characterizing differentiation using aggregate data collected from multiple users and tests. Specifically, we focus on how we detect fixed-rate bandwidth limits, which we refer to as *throttling*. This is by far the

3.5 Dataset

The data generated by Wehe tests includes throughput samples, as well as the following for each back-to-back test: (1) the server timestamp at the beginning of the test, (2) the first three octets (/24) of the client's IP address, (3) the client's mobile carrier as reported by the operating system, (4) the client's operating system and phone model, (5) the network connection type (WiFi or cellular), and (6) the coarse-grained GPS location (collected with user permission). We describe the reason for collecting each of these items below.

The timestamp allows us to identify trends over time. The carrier name allows us to identify the cellular provider for tests on cellular networks. The client's anonymized IP address information and network type allow us to identify the ISP being tested for WiFi connections[2], and to identify whether there are subnet-level differences in detected differentiation.

The coarse-grained GPS location (10 km precision) allows us to identify regional differences in ISPs' policies (e.g., in response to state-level net neutrality regulations in the US). The Wehe app first requests the geolocation of the user via the operating system's precise GPS location feature, the Wehe server then geo-codes the geolocation (i.e., looking up the city/state/country) and stores only the truncated geolocation (i.e., with 10 km precision). Users can choose not to share their GPS locations without limiting app functionality. In 15% of tests, the users opted out of location sharing.

The OS and phone model allow us to distinguish whether ISPs discriminate against these factors, or to what extent OSes and phone models might bias the results.

Summary of dataset We summarize our dataset in Table 1. Between Jan. 18, 2018 and Jan. 24, 2019, 59,326 iOS users and 66,923 Android users installed Wehe and ran at least one test.

In total, Wehe conducted 1,045,413 tests. We plot the distribution of tests over time in Figure 2 (note the log scale on the y-axis). We observe a peak of 77,000 tests on January 19, 2018, when a news article raised awareness of the app [3]. There were three other press events that raised awareness of the app; we still observe several hundred tests per day. Wehe users come from at least 183 countries based on geolocation.

Like any crowdsourced dataset, ours is subject to several biases that may impact the generality of our findings. We cannot control when, where, or why users run our tests, and thus we do not have uniform or complete coverage of any ISP or app tested. Figure 1 shows the distribution of test locations, where the intensity of the color for each country reflects the number of tests completed in the country. More than 60% of our tests come from the US, most likely due to the recent changes in net neutrality rules combined with US-centric press articles. The phone models used in our tests skew toward higher-end devices, Table 2 shows the top phone models and OSes for users in the Wehe dataset. A large fraction of our US tests come from large cellular providers, meaning lower-cost providers (e.g., MVNOs) are under-represented.

Despite these biases, our analysis covers 2,735 ISPs[3] in 183 countries, and identifies differentiation in 30 ISPs in 7 countries. We believe this to be the largest study of content-based differentiation practices.

[2] Using the "OrgName" field from whois queries to regional Internet registries.

[3] We noticed that some ISPs used multiple "OrgNames" (e.g., Bouygues and BouyguesTelecom); thus, some ISPs may be counted multiple times.

Replay	Users (%)	Cellular Tests	WiFi Tests
YouTube	106,813 (85%)	97,009	149,850
Netflix	83,369 (66%)	66,320	112,473
Amazon	77,212 (61%)	61,851	102,529
Spotify	65,644 (52%)	43,306	90,963
Skype	60,658 (48%)	37,589	72,250
Vimeo	49,701 (39%)	33,538	67,333
NBC Sports	49,605 (39%)	38,701	71,701
Total	126,249	378,314	667,099

Table 1: Overview of Wehe data analyzed in this paper.

	iOS		Android	
Users	59,326		66,923	
Top five OS versions	iOS 11.2.2	15%	Android 7.0	17%
	iOS 11.2.5	7%	Android 8.0.0	9%
	iOS 12.1	5%	Android 8.1.0	8%
	iOS 11.4.1	4%	Android 7.1.1	5%
	iOS 11.2.6	3%	Android 6.0.1	4%
Top five phone models	iPhone X	19%	Pixel 2 XL	2.2%
	iPhone 7	14%	Samsung S8	1.9%
	iPhone 6s	12%	Pixel XL	1.8%
	iPhone 7 Plus	11%	Samsung S8+	1.8%
	iPhone 6	7%	Pixel	1.7%

Table 2: Summary of Wehe users' phone models. There is a bias toward newer phones and OSes, with devices capable of displaying content in HD.

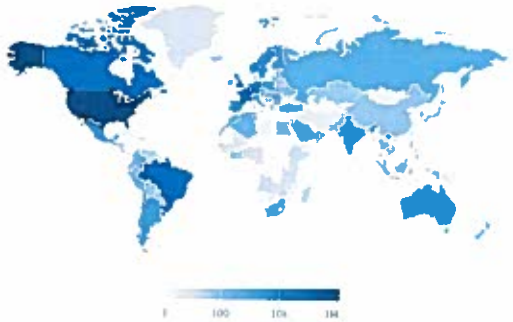

Figure 1: Number of tests per country (log scale). Note that 15% of our tests do not have GPS data (e.g., if the user did not provide permission to collect GPS locations), and we excluded them from any geolocation-based analysis.

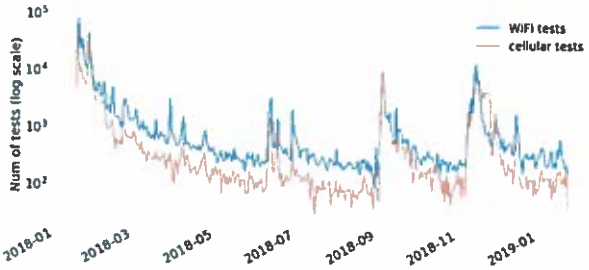

Figure 2: Number of Wehe tests per day (log scale).

4 DETECTING DIFFERENTIATION

We now describe our methodology for identifying and characterizing differentiation using aggregate data collected from multiple users and tests. Specifically, we focus on how we detect fixed-rate bandwidth limits, which we refer to as *throttling*. This is by far the

most common type of differentiation that we observed, and the rest of the paper focuses exclusively on fixed-rate throttling.

Our approach relies on the following steps. Similar to prior work, we use the KS test statistic to detect differentiation by comparing throughput distributions for a collection of original replays to those from control replays [16] (§4.1). For replays where differentiation is detected, we detect one or more throttling rates using kernel density estimation (KDE), under the assumption that throughput samples from clients throttled at the same rate will cluster around this value (§4.2).

Using this approach to detect throttling rates works well if an entire replay is throttled; however, we find in practice that certain devices enforce fixed-rate throttling only after a burst of packets pass unthrottled, as previously reported by Flach et al [13]. We use change point detection on throughput timeseries data to identify delayed throttling periods (e.g., if they are based on time or number of bytes) and omit unthrottled samples when determining the throttling rate (§4.3).

4.1 Identifying differentiation

When identifying differentiation using crowdsourced data, we group tests according to the ISP and the app being tested (e.g., YouTube, Netflix, etc.), which we refer to as an ISP-app pair. We use *all* tests for a given ISP-app pair, where each test consists of one original replay and one bit-inverted replay regardless of whether throttling was detected individually. We focus on ISPs with enough tests to apply the detection methodology; namely, we conservatively require 100 total tests or 10 tests where Wehe identified differentiation.[4] In total, 144 ISPs meet the criteria.

Our null hypothesis is that there is no differentiation for an ISP–app pair. If this is the case, the distribution of throughput samples observed for original and bit-inverted replays should be similar. To test this, we form two distributions: O is the collection of all throughput samples for all original replays for the ISP-app pair and I is the collection of all throughput samples for all bit-inverted replays for the ISP-app pair. Note that the number of samples in O and I are identical by construction (we include only complete pairs of back-to-back replays).

We then test whether O and I are drawn from different distributions by using the Jackknife re-sampling KS Test described earlier. Specifically, we reject the null hypothesis if the KS-Test indicates different distributions with a p-value is 0.05 or less, and the random subsamples of the distribution yield the same result 95% or more of the time.

By aggregating large numbers of tests, we can mitigate the impact of confounding factors such as (random) network dynamics, which should affect both distributions roughly equally given the large number of samples we examine. If we detect differentiation for an ISP-app pair, we next determine whether there is fixed-rate throttling for the pair.

4.2 Inferring throttling rates

The technique we use to detect fixed-rate throttling for an ISP–app pair is based on the hypothesis that when an ISP deploys content-specific fixed-rate throttling, this policy affects multiple users (e.g., those with the same data plan). If this occurs, we expect that multiple tests would be throttled in the same way, and thus the distribution of average throughputs for these tests would be centered at the throttling rate instead of being randomly distributed across the range of available bandwidth for a network.

To detect when average throughputs group around a given rate, we use kernel density estimation (KDE), which estimates the probability density function (PDF) of random variables (in our case, throughput). The intuition behind using KDE is that if the random variable (throughput) contains many samples at or near a certain value, the value should have a probability density that is relatively large. Thus, fixed-rate throttling should lead to relatively large probability densities at or near the throttling rate when using KDE. Note that KDE analysis may yield a PDF that has multiple local maxima, meaning the approach can be used to detect multiple throttling rates (or access technology limits).

There are two key challenges for using KDE effectively to identify fixed-rate throttling. First, we must determine what thresholds to use for identifying local maxima in the PDF that correspond to fixed-rate throttling. Second, we must eliminate confounding factors such as rate limits that are not based on the content of network traffic.

Setting thresholds for detection For the first challenge, we use the following heuristic. We assume that at least some fraction f of the total throughput averages, n, for an ISP-app pair are at the throttling rate, and f represents our detection threshold (i.e., we can detect fixed rate throttling affecting at least $f*n$ tests). We then use an *approximation* that the remaining (i.e., unthrottled) samples are randomly distributed across the available bandwidth for the ISP.[5] Finally, we generate data according to this model, run KDE (using a Gaussian kernel with a bandwidth of 0.1), determine the density for the f throttled samples and use that as our detection threshold t.

More specifically, for each ISP–app pair we find the number of replays n and the average throughput range $[x, y]$. We then construct a distribution consisting of $(1-f) * n$ data points with values uniformly distributed between x and y, and $f * n$ data points with the value $(y-x)/2$. We run KDE on this distribution, and set our detection threshold t to the density value at $(y-x)/2$ (containing a fraction f of the values). We evaluated the methodology with f=0.02 in §5, and we found no false positives or negatives.

Eliminating confounding factors The heuristic above identifies characteristic throughput values containing more samples than would be expected from a uniformly random distribution; however, not all such values are due to fixed-rate throttling. For example, an ISP may impose rate limits on *all* traffic for a device (e.g., due to usage or access-technology limits). Importantly, such behavior should impact *both* the original replays and the bit-inverted replays.

To eliminate such cases, we first remove from consideration any average throughput values that have high density in *both* the original and bit-inverted distributions. Next, we include only throughput values with high density and that correspond to throttling rates observed by Wehe tests that indicated differentiation to the user. For this, we run the same KDE analysis described above, but only on tests where the Wehe app identified differentiation.

[4] These threshold were picked because they avoided false positives for detecting differentiation.

[5] This is not true in practice, but serves as a useful first-order approximation to identify throughput values of interest.

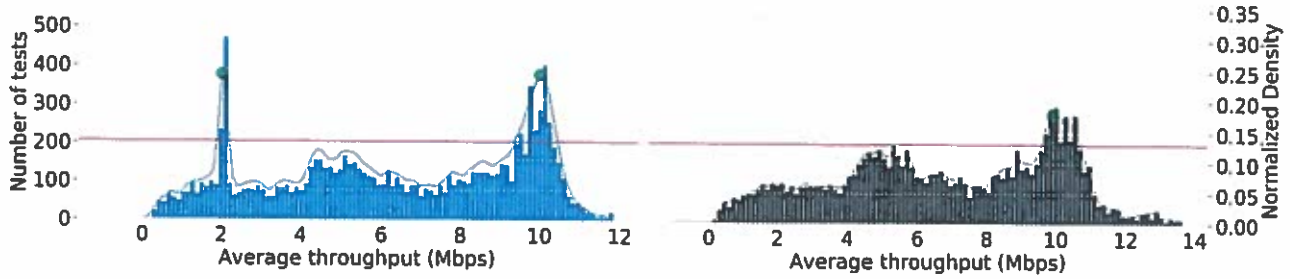

Figure 3: Identification of throttling rate. The x-axis is the average throughput, and the y-axes are a histogram of tests (bars) and probability density function (PDF, gray curve) of average throughputs for all YouTube original replays (left) and all YouTube bit-inverted replays (right) from all tests in Sprint network. The horizontal line is the density threshold for detecting potential throttling rates, with green dots are the values above the threshold. We remove values that appear in both original and bit-inverted, leaving 2.0 Mbps as the detected throttling rate.

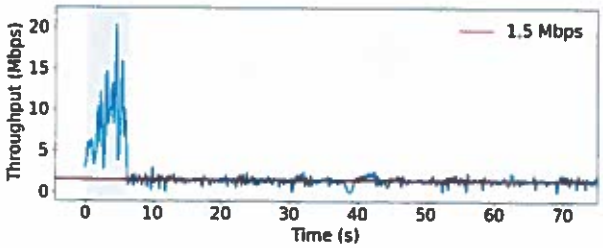

Figure 4: Throughput over time for a Netflix test over T-Mobile, showing delayed throttling. Note that the first few seconds of the transfer include rates up to 20 Mbps, after which they drop to 1.5 Mbps (horizontal line).

As an example of this approach, the left plot in Figure 3 shows a histogram of average throughput values for all YouTube *original replays* over Sprint, and the estimated PDF (grey curve) from running KDE. The horizontal line indicates our detection threshold, t, which identifies high-density values near 2 Mbps and 10 Mbps. The right figure plots the same, but for the *bit-inverted replays*; note that both original and bit-inverted distributions have above-threshold density values at 10 Mbps, indicating that this throughput value is not due to content-based differentiation. Finally, we confirm that tests where the Wehe app indicated differentiation exhibited throttling at 2 Mbps using KDE analysis, and conclude that 2 Mbps is the throttling rate for this ISP-app pair.

4.3 Accounting for delayed throttling

The methods described so far in this section assume that if fixed-rate throttling occurs, it affects the entirety of a Wehe test experiencing throttling. In the case of T-Mobile, we found empirically that this assumption was violated because they engage in *delayed throttling*, previously reported by Flach et al. [13]. Figure 4 shows a timeseries of throughput for a Netflix replay that is subject to this policy: initially the transfer achieves throughput up to 20 Mbps; afterward, the transfer drop to 1.5 Mbps (horizontal line).

Previous work found that delayed throttling was implemented by limiting the number of bytes that are unthrottled, and identified the behavior using the number of bytes that are transferred before the first packet is dropped [13]. In our work, we seek to avoid assumptions about whether such delayed throttling is based on bytes or time, and to use techniques that are insensitive to packet drops caused by reasons other than delayed throttling. Instead, we assume that a detectable delayed throttling session will have at least one phase change, and that all tests for an ISP-app pair affected by delayed throttling will experience the same delay (i.e., number of seconds or bytes). Thus, to detect delayed throttling for an ISP-app pair, we use change point detection (to identify the phase change) and KDE to identify whether the change occurs after a number of seconds or bytes.

Our null hypothesis is that there is no delayed throttling. If this were true, a phase change could be caused by reasons such as bandwidth volatility, and we would expect that the delay would be randomly distributed. To test this hypothesis, we investigate only tests for an ISP-app pair with exactly one phase change, and determine the distribution of delays.

To detect phase changes, we use the PELT algorithm [19] and filter out any tests that do not have exactly one change point. We tuned the detection algorithm so that it would detect change points from tests where we replayed Netflix on T-Mobile's network using our lab devices. To determine whether the change point indicates statistically significant throughput on either side of the boundary, we use a KS test to compare the distributions of throughput before and after the change point. If they are different, we add the change point time and bytes to the list of change points for the ISP-app pair.

After gathering lists of change points, we use KDE[6] to determine whether the change points for the ISP-app pair are randomly distributed or instead cluster together around a time or number of bytes. If there is a relatively large density value at a given number of bytes or time, then we reject the null hypothesis and flag the ISP-app pair as experiencing delayed throttling, according to bytes or time, whichever has the largest density value. As an example, Fig. 5 shows the distribution and estimated PDF of delayed throttling *bytes* for Netflix on T-Mobile, where most of the change points are detected around 7 MB.[7]

If delayed throttling is detected, we filter out throughput samples during the delay and detect the throttling rate as described in the previous section.

4.4 Limitations and Caveats

The methodology for detecting fixed-rate throttling presented in this paper is subject to the following limitations.

[6] With an empirically derived threshold density of 0.1.
[7] The change point times have substantially lower density.

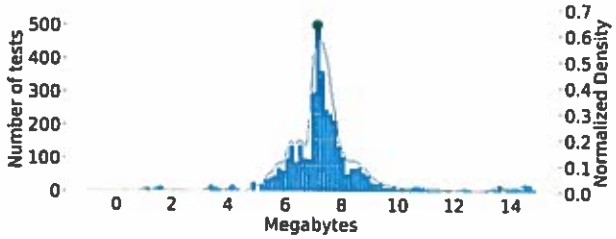

Figure 5: Detecting delayed throttling bytes for Netflix in T-Mobile. For each change point (in bytes) on the x-axis, the figure shows a histogram and estimated PDF generated from KDE. The green dot (at 7 MB) indicates the detected number bytes before throttling begins.

Record/replay limitations The recorded traffic that we use for an app in Wehe's replay tests may not always match the traffic generated by the app. For example, if a video provider switches from HTTP to HTTPS, our tests would be out of date until we create a new recording. Likewise, a throttling device may update its rules for detecting traffic before we deploy a new recording, and this could lead to false negatives. We periodically check for changes to apps that we test in Wehe, e.g., we updated our recordings in mid-January, 2019 after Amazon Prime Video changed from using HTTPS to HTTP.

Detection limits We can find evidence of fixed-rate throttling only when we have sufficient tests (and a sufficient fraction of tests being throttled to the same rate) from an ISP to obtain statistical significance. We detected differentiation for 39 ISPs, but we see no evidence of fixed-rate throttling for 9 of them. Specifically, for these 9 cases we found differences between original and bit-inverted average throughputs, but we did not detect fixed-rate throttling after running KDE. We do not know the root causes for these cases.

5 EVALUATION OF DETECTION METHOD

We now evaluate our detection method using controlled experiments from the four largest US cellular providers. Ideally, we would compare our detection results with ground-truth information from each ISP in our study, but gaining access to each network in our crowdsourced data would be infeasible. Further, even if we had this information, we could not control for confounding factors such as varying network conditions, the user's data plan or usage history.

Instead, we validate that our detection methodology produces findings that are *consistent* with controlled experiments performed in our lab. For the largest four US carriers, we do find consistent results—our lab tests indicate content-based differentiation and fixed-rate throttling that matches results produced by our analysis of data from Wehe users.

5.1 Lab experiment setup

We purchased SIM cards from AT&T, Sprint, T-Mobile and Verizon. We intentionally purchased prepaid plans that mention indicators of throttling practices, such as "video streaming at 480p" or "video optimized streaming." Note that none of the disclosures indicated which video providers are targeted for throttling, nor how the targeting is done. We conducted lab experiments in Jan. 2018, May 2018 and Jan. 2019 for AT&T, T-Mobile and Verizon, and the tests for Sprint only in January, 2019 due to difficulty acquiring a prepaid SIM.

For each experiment, we ran each of the 7 Wehe tests on each SIM card 10 times. We include two sets of tests for Vimeo (with two different domains) and Amazon Prime Video (one using HTTPS and one using HTTP) in Jan. 2019 to reflect the change in how the service delivered video that month.

Since the data plan disclosures did not indicate which video services were throttled, we do not have ground truth for which Wehe tests should be affected. Instead, our hypothesis is that if our lab tests are affected by content-based differentiation, then we should be able to detect exactly which content triggers throttling. We use the "binary randomization" method [20] for identifying content that triggers DPI classification rules used in throttling deployments

5.2 Comparison with Wehe data

To compare the lab findings with crowdsourced Wehe data, we build subsets of Wehe data, one each from Jan., 2018 and May 2018, and two from Jan. 2019 to reflect updated recordings released that month. We then use the methodology from the previous section to detect fixed rate-throttling and compare our findings with those from lab experiments. Additional findings from our lab setting are discussed in Appendix A.

Table 3 presents a summary of findings, showing that our lab tests and crowdsourced data are consistent. There are at least three columns for each ISP-app pair, representing tests from Jan., 2018, May 2018 and Jan., 2019. There is an additional column for Amazon and Vimeo where we separate out the tests based on whether they were done using older (the third column) or newer traces (the fourth column). A shaded cell indicates that our method detected differentiation using crowdsourced tests for that ISP-app pair from that specific month, while a white cell means that we did not. A ✓ shows that the result from Wehe data matches the lab experiment for an ISP-app pair during that month, and a "-" indicates cases where we have no lab experiments (January/May 2018 for Sprint).

Table 3 shows that all cases of throttling in lab experiments were also detected in Wehe tests. We could not verify consistency for all Wehe crowdsourced findings; namely, our tests indicate throttling of Skype video in the first nine months of 2018, but we did not have a Sprint SIM for lab tests then.

6 CHARACTERIZING DIFFERENTIATION

We now present our findings from all Wehe tests in our dataset. In this section, we focus on cases where throttling is detected for at least one ISP-app pair. Table 4 summarizes the results. Additional detail of the findings in Table 4 are presented in Appendix C. While the majority of *tests* come from WiFi networks, the majority of *detected differentiation* occurs in cellular networks. We discuss our findings in more detail below.

6.1 Identified differentiation

We identified 30 ISPs in 7 countries that throttle at least one Wehe test. Nearly all cases of detected throttling affect video streaming services, with YouTube being throttled the most often (25 cases), and Vimeo being throttled the least (3 cases).