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
© 2019 Association for Computing Machinery.
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, N for Netflix, a for Amazon Prime Video, 🦚 for NBCSports, S for Skype, ● for Spotify and v 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.

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.

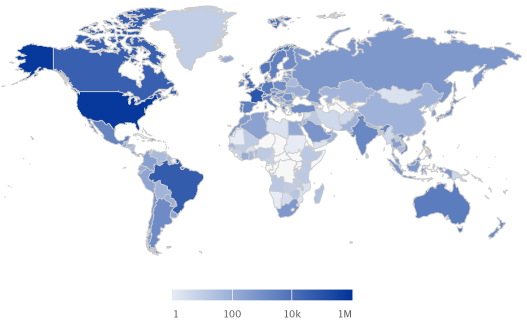


Figure 1: Number of tests per country (log scale). Note that 15% of our tests do not have GPS data (e.g., if the user did not provide permission to collect GPS locations), and we excluded them from any geolocation-based analysis.

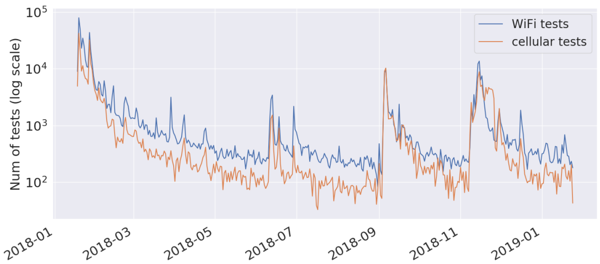


Figure 2: Number of Wehe tests per day (log scale).

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

[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.

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.

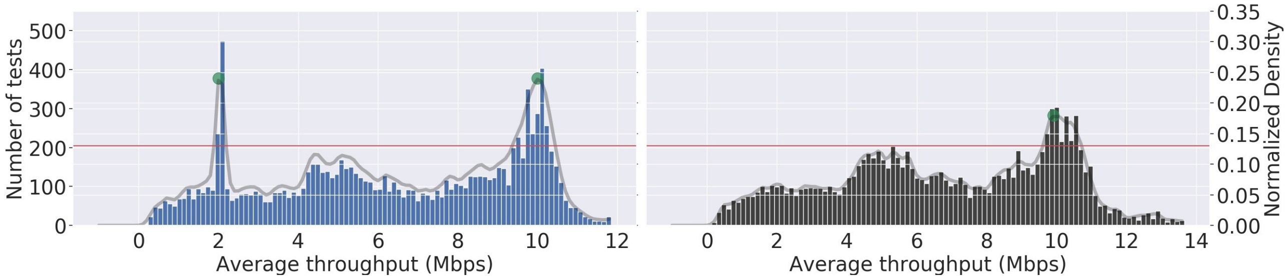


Figure 3: Identification of throttling rate. The x-axis is the average throughput, and the y-axes are a histogram of tests (bars) and probability density function (PDF, gray curve) of average throughputs for all YouTube original replays (left) and all YouTube bit-inverted replays (right) from all tests 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.

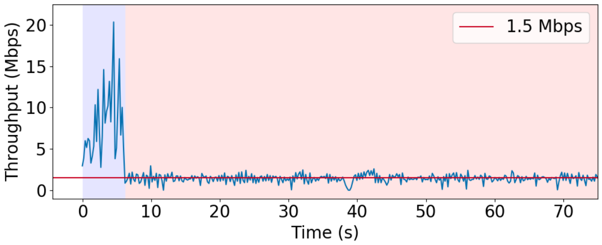


Figure 4: Throughput over time for a Netflix test over T-Mobile, showing delayed throttling. Note that the first few seconds of the transfer include rates up to 20 Mbps, after which they drop to 1.5 Mbps (horizontal line).

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.

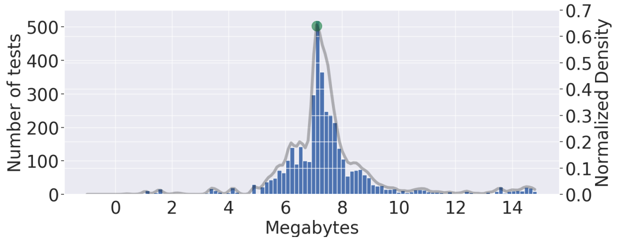


Figure 5: Detecting delayed throttling bytes for Netflix in T-Mobile. For each change point (in bytes) on the x-axis, the figure shows a histogram and estimated PDF generated from KDE. The green dot (at 7 MB) indicates the detected number bytes before throttling begins.

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).

	▶			N			a				🦚			v				S			◉		
	01/18	05/18	01/19	01/18	05/18	01/19	01/18	05/18	01/19	01/19′	01/18	05/18	01/19	01/18	05/18	01/19	01/19′	01/18	05/18	01/19	01/18	05/18	01/19
AT&T	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	✓	✓	✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Verizon	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
T-Mobile	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	▨ ✓	✓	✓	▨ ✓	✓	✓	✓	✓	✓	✓	✓
Sprint	▨ -	-	▨ ✓	▨ -	-	▨ ✓	▨ -	-	▨ ✓	▨ ✓	-	✓	✓	-	-	✓	✓	▨ -	-	✓	-	-	✓

Table 3: Comparison of crowdsourced and lab results on four US mobile carriers in Jan. 2018, May 2018, and Jan. 2019. There is an additional column (') for new Amazon and Vimeo traces recorded in Jan. 2019. A shaded box indicate throttling was detected using crowdsourced data; otherwise throttling was not detected. A ✓ indicates that result from crowdsourced data matched the lab experiment results, and "-" means we do not have a lab experiment.

Our methodology did not detect any ISP throttling of Spotify tests in our data, and detected throttling of Skype video tests only in Sprint (§6.3), Boost Mobile (which is owned by Sprint), and the United Arab Emirates (UAE) on both WiFi and cellular connection. In the UAE, the "throttling" is to zero (i.e., Skype tests are blocked), reportedly because Skype provides an unlicensed VoIP service in the country.

The most common detected throttling rate is 1.5 Mbps (12 cases). These rates typically correspond to ISPs that disclose data plans offering low-resolution video streaming, a topic we investigate in §7. Besides blocking, the lowest throttling rate detected is 0.5 Mbps from Boost Mobile, and the highest detected throttling rate is 6 Mbps from Start Communications, a regional ISP based in Ontario, Canada.

Throttling via WiFi In the vast majority of ISPs tested via WiFi, our methodology did not detect throttling. The exceptions were the UAE blocking Skype (1 instance), and five other providers in North America. At least three of these five (Viasat, Hughes and NextLink) are satellite providers, and are likely more bandwidth constrained. While we cannot confirm the type of network for VianetTV and Start Communications, Vianet's website indicates that they offer residential plans that carry Internet traffic over cellular connections. Thus, the majority of detected throttling over WiFi occurred in networks that carry traffic over long-range wireless networks.

Throttling over cellular connections Most cellular throttling comes from providers in the US. We detected differentiation in *nearly every* major US cellular ISP, and we found all these throttling practices started before June 2018 (i.e., when the FCC rolled back restrictions on throttling [4]). While the number of detected cases in the US might be due to the bias in our dataset, it is in part due to the regulatory regime. For example, we do not detect throttling from cellular ISPs in France, where we have a large sample size, and where the practice is illegal. One notable exception is Google Fi, which did not throttle any of our tests.

6.2 Variations in detected throttling

Not all tests for each ISP-app pair in Table 4 are throttled. We now investigate several potential root causes for the behavior.

Policy changes over time One explanation for non-uniform throttling of Wehe tests is that throttling policies changed for ISP-app pairs during our study. We test this by comparing the detected throttling rates and fraction of throttled tests over time. The number of tests per ISP-app pair varies considerably over time, so we use the following approach to test sample sizes with sufficient power to draw conclusions about policy changes. For each ISP-app pair, we divide tests into periods, each with a minimum of 100 total tests and 10 throttled tests. If the same rate(s) and a similar fraction of throttled tests are observed in all periods, we conclude there is no policy change over time. We consider only two periods that meet this criteria: periods of one month, or periods of 6 months. The latter divides data into periods before and after the US rolled back net neutrality protections in June, 2018.

Country	ISP	Throttled Apps	Rate(s)	Tests
		WiFi network		
Canada	Start Comms.	a, 🦚	6 Mbps	126
Canada	ViaNetTV	▶, N	1 Mbps	45
UAE	Etisalat	S	0 Mbps	23
US	Hughes Net. Sys.	▶, N	1 Mbps	81
US	NextLink	▶, N, a, 🦚, v	4 Mbps	72
US	ViaSat	▶, N	1 Mbps	112
		Cellular network		
Canada	Rogers	▶, N	1.5 Mbps	4479
Canada	SaskTel	N	1.5/3 Mbps	61
Chile	Entel	▶	1.5 Mbps	30
Germany	Telekom DE	▶, a	1.5 Mbps	178
Israel	HOTmobile	▶	1.5 Mbps	23
UAE	Etisalat	S	0 Mbps	73
UAE	du	S	0 Mbps	44
US	AT&T	▶, N, 🦚	1.5 Mbps	46,013
US	BoostMobile	▶, N, a, S	0.5/2 Mbps	792
US	Cellcom	▶, N, a	4 Mbps	97
US	Cricket	▶, a	1.5 Mbps	1,224
US	CSpire	▶, N	1 Mbps	41
US	FamilyMobile	▶, N, a, 🦚	1.5 Mbps	106
US	GCI	▶, N, a, 🦚, v	1/2 Mbps	153
US	Iowa/iWireless	▶, N, a, 🦚	1.5/3 Mbps	76
US	MetroPCS	▶, N, a, 🦚	1.5 Mbps	2,135
US	Sprint	▶, N, a, S	2 Mbps	35,295
US	T-Mobile	▶, N, a, 🦚, v	1.5 Mbps (delayed)	39,820
US	Tracfone Wireless	▶, a	2 Mbps	410
US	Verizon	▶, N, a	2/4 Mbps	69,016
US	Visible	▶, N	2 Mbps	52
US	XfinityMobile	▶, N, a	2 Mbps	131
UK	giffgaff	▶, N	1 Mbps	58
UK	O2	▶, N	1.5 Mbps	210

Table 4: ISPs where we detect differentiation, the throttling rates, apps affected, and the number of Wehe tests in the ISP. A more detailed version is in Table 7.

The monthly analysis covers AT&T, Sprint, T-Mobile, Verizon, and MetroPCS, and the biannual analysis covers BoostMobile, cricket, O2, and Tracfone Wireless. For the vast majority of cases, we see the same throttling rate and similar fractions of throttled tests during the study period, indicating that most policies are stable over the course of one year, and that the throttling policies were in place even before new FCC rules permitted them in June, 2018.

For T-Mobile, we detected that Vimeo tests are throttled only after Nov., 2018, and a small fraction of YouTube tests were throttled at 2 Mbps (instead of 1.5 Mbps) only in Jan. 2019. For Sprint, we found that Skype video tests ceased to be throttled after Oct., 2018. We detected no changes in other policies.

Time-of-day We now investigate the role time-of-day plays in throttling, in light of claims that throttling is necessary to prevent overloading the network during times of heavy usage [7]. If this were true, we expect to see higher incidence of throttling during a cellular network's busy hours (e.g., 8am to midnight) compared to overnight (e.g., midnight to 8am). We test the hypothesis by grouping tests into day and night using busy hours identified in prior work [25], and checking whether the fractions of the throttled tests are different (e.g., more tests are being throttled during the day).

Specifically, for each ISP-app pair we denote the fraction of throttled tests as f, number of total daytime tests D (d of which are throttled), and N total nighttime tests (n of which are throttled). If there is no time-of-day effect, the number of throttled tests should be $D * f$ during the day and $N * f$ during the night. We run a chi-squared test comparing the actual number of throttled tests (n and d) with the expected number of throttled tests (D^*f and N^*f); if the p-value is less than 0.05, we conclude there is a time-of-day effect.

Out of the 77 ISP-app pairs we detected throttling, 71 of them include tests both during the day and night; of these we find no evidence of time-of-day effect for 60 cases. Of the remaining 11, four have fewer than 30 tests and thus limited statistical power, and we manually investigated the remaining 7 ISP-app pairs. For these cases, we found the opposite result as our hypothesis: the fraction of tests throttled during busy periods decreased compared to non-busy periods. This could be due to a different set of users with different throttling policies, or it could be due to a lack of sufficient available bandwidth to detect throttling. Our data supports the latter explanation, because during busy hours we see a larger fraction of original and bit-inverted tests with throughput lower than the throttling rate.

Geographical differences We investigated whether there are geographic differences in throttling practices, e.g., one region is affected more than another. This could be due to factors such as state-level net neutrality laws or a regional deployment of throttling (e.g., affecting a subset of a provider's packet gateways). We focus on the US-based ISPs (where we have the most samples), and conduct a state-level regional analysis. Our finding is that there are differences in throttling experienced by Wehe users in each state, but these variations are not persistent and are consistent with random noise.

Mobile OSes We consider whether the mobile OS plays a role in whether a given client will be throttled or not. We analyzed the fraction of tests for an ISP-app pair affected by differentiation for iOS and Android, and found that the top four US cellular providers have similar throttling rates for both mobile OSes.

IP prefixes We next consider whether throttling affects only certain IP prefixes assigned to clients. We first grouped the tests according to the routable IP prefix that the client's IP address belongs to, then determined the fraction of throttled tests for each prefix. If differentiation is implemented on a per-prefix basis we would expect a bimodal trend with prefixes having either no cases of throttling, or nearly all tests experiencing throttling. However, this is not what we observe.

For each ISP-app pair, we calculated the fractions of throttled tests for each IP prefix, and then checked the standard deviation of the fractions, if the distribution is bimodal,we would expect a high standard deviation. In more than 87% of the cases, we observe a standard deviation of less than 0.2; we manually checked the remaining cases and did not see a bimodal trend.

Other possible explanations Variations in throttling could be due to ISPs offering different service plans and features, only some of which include throttling. When visiting websites for several ISPs where we detected throttling, we found options to subscribe to (more expensive) plans that did not limit video streaming (e.g., ViaSat's Platinum 100 plan), and/or features to disable throttling (e.g., AT&T's Stream Saver). Because our dataset does not include plan information, we cannot quantify the impact of these factors.

6.3 Case studies

We now present several notable findings using our methodology on crowdsourced Wehe data from the top four US carriers. A common case is presented in Fig. 6(a), depicting CDF of average throughput for original and bit-inverted YouTube replays. The vast majority of original samples cluster around 1.5 Mbps (the detected throttling rate) while the bit-inverted replays are (mostly) evenly distributed across the throughput range.

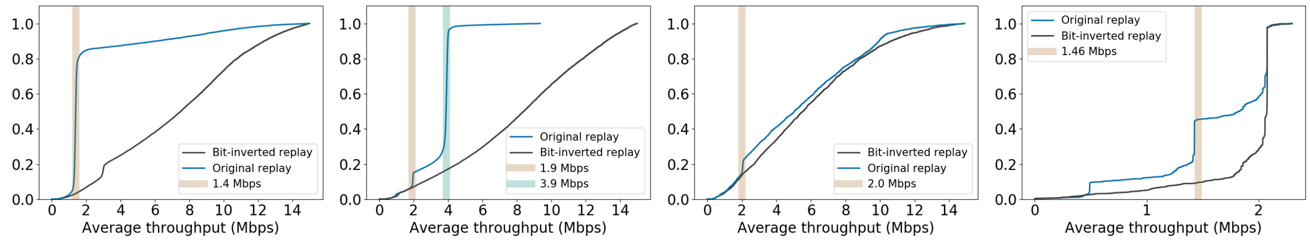
Multiple throttling rates for the same ISP-app pair KDE analysis revealed that Verizon has two throttling rates, one at 4 Mbps (the majority of throttled tests) and the other at 2 Mbps. We show this using a CDF of average throughputs in Fig. 6(b). We believe this is due to different plans offered by Verizon; e.g., in Dec. 2018 their "go unlimited" plan included "DVD-Quality Streaming (480p)" while their "beyond unlimited" plan allowed "HD-Quality Streaming (720p)" [5].

Small fraction of tests affected by throttling Our methodology identifies throttling in Sprint, despite a small percent (4.8%) of tests being affected. To demonstrate this visually, we plot a CDF of average throughput samples for original and bit-inverted replays in Fig. 6(c). There is an inflection point at 2 Mbps, the detected throttling rate, which we also detected in lab experiments using a prepaid SIM. We suspect the reason for such a small fraction of tests being affected is that throttling happens on uncommon data plans, such as the prepaid one we used for lab experiments.

Different policies for different video-streaming apps As discussed in §4.3, T-Mobile implements delayed throttling based on bytes. Interestingly, we find that not all throttled video streaming services are treated equally under this policy. We detected 7 MB of delayed throttling for Netflix and NBC Sports, and 6 MB for Amazon Prime. YouTube does not get any delayed throttling in our dataset—they are throttled from the start.

Skype tests in Sprint We did not detect throttling of Skype video calls in our lab experiments on a Sprint SIM; however, our methodology found evidence of Skype video throttling from Wehe crowdsourced data. Fig. 6(d) shows a CDF of average throughputs for original and bit-inverted Skype video tests, with detected fixed-rate throttling at 1.5 Mbps. When focusing on Jan., 2018 data, we find that the Jackknife KS test used to detect differentiation has a p-value of $8 * 10^{-94}$ with 100% accept ratio—strong evidence of throttling.

Interestingly, Wehe tests identified such differentiation until Sep. 2018, but the tests no longer indicated differentiation afterward. One explanation for this behavior is that Sprint no longer throttles based on content in our Skype video tests. When asked (in Oct., 2018) to comment on our findings regarding Skype and other tests indicating throttling, a press spokesperson from Sprint replied: "Sprint does not single out Skype or any individual content provider in this way." Our lab tests in 2019 corroborate the claim about Skype (but does not speak to the early 2018 findings); however, our lab



(a) YouTube tests on AT&T (all data)

(b) YouTube tests on Verizon (all data)

(c) YouTube tests on Sprint (January 2018)

(d) Skype tests on Sprint (January 2018)

Figure 6: CDF of average throughputs from YouTube ((a)–(c)) and Skype (d) tests. For AT&T, the detected throttling rate is 1.5 Mbps in 6(a), for Verizon there are two detected rates (2 Mbps and 4 Mbps) in 6(b). In Sprint, we detect throttling of a small portion (4.8%) of the original replays throttled to 2 Mbps in 6(c). For Skype tests on Sprint in 6(d), the detected throttling rate is 1.5 Mbps.

tests also identify that Sprint *does* single out content providers via content-based throttling.

7 IMPACT ON VIDEO STREAMING

While Wehe enables us to observe differentiation at scale, it does not provide details about the video resolution for a given app (e.g., which video resolutions are selected by a video streaming app). As described in Section 3, Wehe simply replays the payloads recorded from video streaming, and does not adapt bitrates dynamically. To address this, we need additional experiments to help us understand how streaming apps (with adaptive bitrate) behave when being throttled.

This section describes how we conduct these additional measurements by instrumenting video streaming apps to determine how throttling impacts the video resolution selected by each player. We focus on this metric because it is the one most often cited in ISPs' throttling disclosures (e.g., "video streams up to 480p"), but to date has received little attention from auditing measurements. We first describe the data collected in Section 7.1. We then discuss the impact of throttling and app's data usage setting on streaming in Section 7.2. Finally, we identify root causes for observed behavior using sequence-time diagrams in Section 7.3.

7.1 Measuring video resolution

Experiment environment We analyze Netflix, YouTube and Amazon on prepaid plans from AT&T[8], T-Mobile, Verizon, and Sprint between Jan. 14 and Jan. 25, 2019. We present the impact of throttling on video quality and throughput over each ISP, and compare each result with tests (1) over WiFi, (2) when connected via an encrypted VPN on the same cellular connection, and (3) when disabling any data-saving by the apps (i.e., enabling streaming at the maximum rate according to the app). The WiFi network is not throttled and VPN tunnels evade any content-based throttling. In each network setting, we use each app to stream video for two minutes.[9] We repeat each ISP-app experiment five times and present summary results.

Video streaming In each video streaming session, we stream the same video and let the client app determine what video resolution to use. While the bitrate selection code is unavailable to us, we expect that the video streaming session is influenced by factors such as encoded bitrates, network conditions, access technology, and data usage settings. We discuss the factors that we vary in our tests.

To vary whether a video streaming session is affected by content-based throttling, we stream video with and without an encrypted VPN tunnel. For access technology, we run tests over both cellular and WiFi connections. For the WiFi network, we confirm that neither the network nor the app (the app does not attempt to save data on WiFi) are the bottleneck. Finally, for the data usage settings, we note that Netflix and Amazon provide an option in their app to let users control how much data they want to use over cellular connections. Amazon has three settings: Good (0.6GB/hour), Better (1.8GB/hour) and Best (5.8GB/hour). Netflix also provides three settings Automatic, Save Data and Max data which allow 4 hours, 6 hours and 20 minutes per GB. For these apps we test both their default[10] and most data-intensive settings (which we refer to as "Max data").

Video quality and throughput There are no publicly known APIs to collect video quality information from mobile apps. Our approach is to monitor video quality during playback using app features that print video resolution on the screen, then use optical character recognition (OCR) to extract them for YouTube and Netflix. Amazon does not expose video resolution information, so we obtain this data from the HTTP GET request and the manifest file. We calculate the throughput based on packet capture data.

7.2 Impact of throttling

We begin by analyzing the impact of throttling on video streaming resolution and throughput.

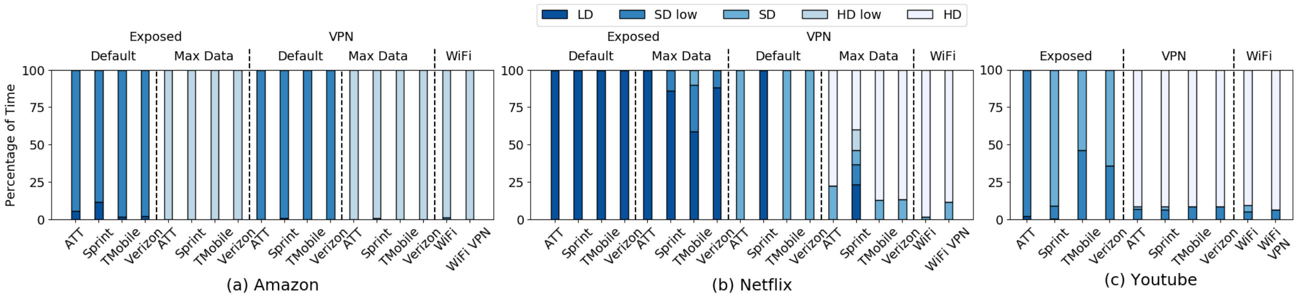
Throttling decreases playback resolution Figure 7 shows the percentage of time the video is streamed at each resolution. The precise resolutions and their mappings are listed in Appendix B. Each subfigure plots the results for a different video streaming service, and each plot groups our results according to whether the test exposes the original packet payloads ("exposed") or uses a VPN to conceal them. As expected, in tests where packet payload is exposed for Netflix and YouTube, the playback resolution is lower than the cases where the VPN conceals the payload. This result holds even when we turn off any data saving mechanisms ("Max data" and "VPN Max data"). The exceptions are Amazon and Sprint on Netflix. We discuss the Amazon case below, but do not have a root cause for Sprint/Netflix.

Cellular networks can support higher throughputs Figure 8 presents the throughput observed while streaming under the default app settings for exposed (solid) and tunneled traffic (dashed). We

[8] The AT&T SIM had Stream Saver [1], which throttles video traffic, enabled by default.

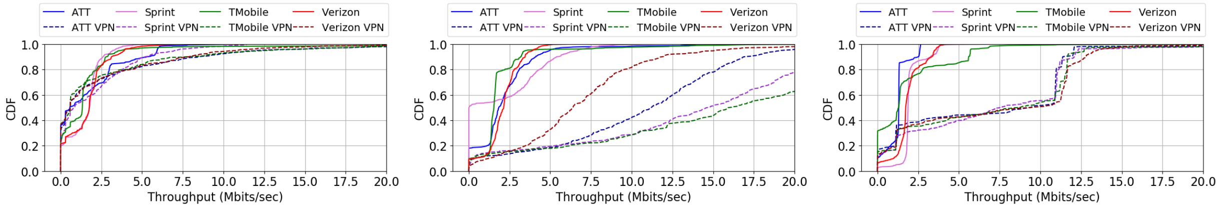
[9] We used iPhones (6S and 8) and a Nexus 6 and obtained similar results from all three.

[10] "Automatic" for Netflix and "Good" data usage on Amazon.



(a) Amazon (b) Netflix (c) Youtube

Figure 7: Stacked histogram for each video streaming service, showing the percentage of time the video is streamed at each resolution (LD, SD, and HD are low, standard, and high definition). The precise resolutions are in Table 6.



(a) Amazon (b) Netflix (c) YouTube

Figure 8: CDF of throughput for each video streaming service (with low data usage settings) and each carrier.

confirmed there was sufficient cellular bandwidth (using Speedtest) of at least 20 Mbps in all tests. This shows that cellular networks support much higher throughput than the throttling rate (as indicated by the larger average throughputs for VPN curves of Netflix and YouTube). The exception is Amazon, discussed below.

Apps default to limiting their streaming rates We find that Amazon and Netflix, by default, use a lower video resolution than the network can support, with or without the VPN (Fig. 7). When compared with "Max data," nearly all of the tests using the default data usage setting select video resolutions that were below 480p (SD), with Netflix picking a resolution as low as 384x216 (LD) and Amazon picking 710x296 ("SD low"). These are substantially lower than the phone screen resolution (1334x750). When we disable the default behavior and allow the apps to stream at their highest achievable rate, video streaming services are able to achieve significantly higher resolutions—indicating that, except for Netflix on Sprint, the cellular networks tested have sufficient bandwidth to support HD video.

Amazon over VPN connections is a special case. Unlike others, the throughput does not increase while using a VPN because Amazon's default data usage settings restricts the app to only use 0.6GB per hour, or an average of 1.6 Mbps both with and without the VPN. When we disable the default throughput limitations (not shown) Amazon has throughputs of 2 Mbps when the packet payloads are exposed and throughput of 4.5 Mbps over VPN. Note that the reason Amazon does not appear to be limited by 1.5 Mbps throttling on AT&T is because AT&T throttles each TCP connection to 1.5 Mbps individually, and Amazon uses multiple TCP connections.

In summary, throttling limits maximum video resolution, but apps' default settings and available resolutions also play a significant role.

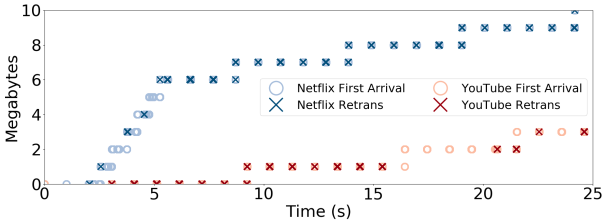


Figure 9: Bytes over time when streaming Netflix (red) and YouTube (blue) on T-Mobile. Netflix experiences delayed throttling, but not YouTube.

7.3 Transport-layer impact of throttling

We now investigate how throttling impacts video streaming at the transport layer. We explore this impact in Figure 9 by considering the bytes transferred over time for each video stream. Each figure is annotated with the initial transmission of a packet (circles) as well as retransmission events (×). We collect packet captures for this analysis from a (non-rooted) iPhone via standard developer tools for iOS [8], and we use the definition of "TCP retransmission" in Wireshark [9].

Transparent proxies and the transport layer We observe AT&T and Verizon implementing TCP terminating proxies in their networks with drastically different results for the transport layer. Though separate analysis with Wehe, we identified that AT&T uses a transparent TCP proxy to split the connection, buffer packets from the server, and pace packets between the proxy and mobile device, at a rate of 1.5 Mbps. This buffering and pacing of packets results in throttling that does not incur high rates of retransmissions.

In contrast to AT&T, the *retransmission rate is 23%* when streaming Netflix on Verizon, the highest among the other carriers and high by any standard. We conducted additional experiments to investigate the root cause of this behavior. Namely, we used Wehe tests in lab experiments and observed the same high retransmission rates at the client; however, the server traces indicated little-to-no retransmission. Thus, we believe that Verizon implements a transparent TCP proxy like AT&T; however, unlike AT&T, Verizon's proxy does not pace packets, instead sending them faster than the throttling device allows (and thus leading to high packet loss). Interestingly, there is minimal impact of the high retransmission rate on video streaming, likely because the video streaming buffer absorbs any transient disruptions to packet transfers.

Policies can differ between applications Figure 9 shows the bytes over time when streaming a video on Netflix and YouTube over T-Mobile's network. Note that when retransmission and first-arrival markers overlap, there are time gaps of 10s of milliseconds, but not visible on a graph (which is measured in 10s of seconds). In each cluster of points, the retransmissions occur first (and correspond to bytes sent one RTO earlier), then as the retransmitted packets are ACKed, new first transmissions occur 10s of milliseconds later.

We observe that T-Mobile throttles Netflix after 7 MB of data transfer (delayed throttling), while it does not delay throttling for YouTube. While packet loss is zero during the delayed throttling period, immediately afterward the retransmission rate is 26%, eventually reducing to 17%. By comparison, YouTube initially experiences a loss rate of 6.8% and drops to 3% after 70 seconds. In both cases, losses waste substantial bandwidth, but the problem is more acute for cases with delayed throttling due to TCP sending at a high rate and adapting slowly.

8 DISCUSSION

This section discusses additional considerations about our findings, their generality, and future work.

Bias towards the US. Most of our data comes from the US, which necessarily biases our findings in a way that likely undercounts differentiation outside the US. In addition, Wehe includes tests for video and music streaming apps, as well as VoIP and videoconference apps, that are popular in the US. However, it is likely that other apps are more popular in other countries, and some of those apps may be throttled. If this is the case, we would underreport the prevalence of throttling in such regions. In the future, we will add tests for more apps that are popular in other regions.

ISPs where throttling was not detected. We gathered sufficient samples to detect differentiation in 144 ISPs, and detected differentiation in 30 of them. This suggests that the majority of ISPs that we studied do not deploy content-based differentiation. Examples include major broadband providers in the US (e.g., Comcast), and all broadband and cellular ISPs in France. Note, however, that some ISPs may throttle using methods other than content-based differentiation (e.g., IP address or monthly data usage) that Wehe cannot detect. Thus, we can only say that we did not detect content-based differentiation, but we cannot tell whether other differentiation occurs.

Ground truth. It is difficult, and in some cases impossible, to find ground truth for every ISP in our study. However, we did validate, via documentation on providers' websites, that throttling policies exist for most US carriers and for several outside the US. That said, there are many ISPs that either do not disclose this information or make it hard to find. There is a clear need for better transparency and more uniform ways of disclosing throttling behavior.

Future of DPI-based differentiation. We identified that the classification rules used by ISPs for throttling rely on plaintext payload contents (e.g., SNI field in TLS handshake). In newer protocols such as TLS 1.3 with encrypted SNI (or QUIC with similar features), such information will no longer be in plaintext—begging the question of how DPI devices will identify traffic for differentiation. We believe that content-based differentiation might still exist even when using such protocols, e.g., by correlating flow IPs with the plaintext names in DNS lookups that they correspond to. Of course this can be addressed by technologies like DNS over HTTPS. Assuming all content is encrypted (even DNS), we envision that classifiers will search for traffic patterns instead of text strings. Because Wehe preserves traffic patterns, we believe our approach will still work.

Complex relationships between content providers, ISPs, and throttling practices. We showed that throttling practices are deployed by many ISPs, and these practices generally worsen performance for content providers in terms of packet loss and decreased video quality. However, we cannot identify the extent to which content providers are (dis)satisfied with such policies. For example, content providers may experience reduced transit costs for throttled video when compared to unthrottled video that uses higher resolution and more bandwidth. It is also possible that ISPs and content providers have entered into agreements to collaboratively control traffic volumes from streaming video. In short, the relationship between content providers, ISPs, and deployed traffic management practices may be more complicated than publicly disclosed. Of course, understanding such relationships is outside of the scope of this work.

9 CONCLUSION

In this work, we conducted a large-scale, one-year study of content-based traffic differentiation policies deployed in operational networks. We developed and evaluated a methodology that combines individually error-prone device measurements to form high-confidence, statistically significant inferences of differentiation practices, and identified differentiation in both cellular and WiFi networks. We found that most throttling targets video streaming, and that there are a wide range of throttling implementations detected in our dataset. In addition, we investigated the impact of throttling on video streaming resolution, finding that while throttling does limit video resolution, it is also the case that default settings in video streaming apps in some cases are the primary reason for low resolution. We are making our code, dataset, and summary of findings publicly available to inform stakeholders and bring empirical data to discussions of net neutrality regulations.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd, Walter Willinger, for their valuable feedback. This work was funded in part by the National Science Foundation (CNS-1617728, CNS-1700657, CNS-1350720, CNS-1740895, and CNS-1651784), a Google Research Award, Verizon Labs, Arcep (Autorité de Régulation des Communications Électroniques et des Postes), and an AWS Research Grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, Google, Arcep, Verizon Labs or Amazon.

REFERENCES

[1] 2017. AT&T Stream Saver. https://www.att.com/offers/streamsaver.html. (April 2017).

[2] 2018. All you need to know about Net Neutrality rules in the EU. https://berec.europa.eu/eng/netneutrality/. (April 2018).

[3] 2018. Apple Is Blocking an App That Detects Net Neutrality Violations From the App Store. https://motherboard.vice.com/en_us/article/j5vn9k/apple-blocking-net-neutrality-app-wehe. (April 2018).

[4] 2018. FCC Releases Restoring Internet Freedom Order. https://www.fcc.gov/fcc-releases-restoring-internet-freedom-order. (January 2018).

[5] 2018. Verizon Unlimited Plans. https://www.verizonwireless.com/plans/unlimited/. (December 2018).

[6] 2018. Wehe: Check Your ISP for Net neutrality Violations. https://dd.meddle.mobi/. (April 2018).

[7] 2019. AT&T Plans. https://www.att.com/plans/wireless.html. (January 2019).

[8] 2019. Recording a packet trace. https://developer.apple.com/documentation/network/recording_a_packet_trace. (June 2019).

[9] 2019. Wireshark's TCP Analysis. https://www.wireshark.org/docs/wsug_html_chunked/ChAdvTCPAnalysis.html. (June 2019).

[10] Vitali Bashko, Nikolay Melnikov, Anuj Sehgal, and Jurgen Schonwalder. 2013. BonaFide: A traffic shaping detection tool for mobile networks. In *In Proc. of Integrated Network Management (IM2013).*

[11] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna P. Gummadi, Ratul Mahajan, and Stefan Saroiu. 2010. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proc. of USENIX NSDI*.

[12] FCC. 2015. Protecting and Promoting the Open Internet. https://www.federalregister.gov/articles/2015/04/13/2015-07841/protecting-and-promoting-the-open-internet. (April 2015).

[13] Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, and Ramesh Govindan. 2016. An Internet-wide analysis of traffic policing. In *Proc. of ACM SIGCOMM*. ACM.

[14] Frank J. Massey Jr. 1951. The Kolmogorov-Smirnov Test for Goodness of Fit. *J. Amer. Statist. Assoc.* 46, 253 (1951).

[15] Arash Molavi Kakhki, Fangfan Li, David R. Choffnes, Ethan Katz-Bassett, and Alan Mislove. 2016. BingeOn Under the Microscope: Understanding T-Mobile's Zero-Rating Implementation. In *Proc. of SIGCOMM Workshop on Internet QoE*.

[16] Arash Molavi Kakhki, Abbas Razaghpanah, Anke Li, Hyungjoon Koo, Rajesh Golani, David R. Choffnes, Phillipa Gill, and Alan Mislove. 2015. Identifying Traffic Differentiation in Mobile Networks. In *Proc. of IMC*.

[17] Partha Kanuparthy and Constantine Dovrolis. 2010. DiffProbe: detecting ISP service discrimination. In *Proc. of IEEE INFOCOM*. IEEE.

[18] Partha Kanuparthy and Constantine Dovrolis. 2011. ShaperProbe: End-to-end Detection of ISP Traffic Shaping using Active Methods. In *Proc. of IMC*.

[19] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. 2012. Optimal detection of changepoints with a linear computational cost. *J. Amer. Statist. Assoc.* 500 (2012).

[20] Fangfan Li, Arash Molavi Kakhki, David Choffnes, Phillipa Gill, and Alan Mislove. 2016. Classifiers Unclassified: An Efficient Approach to Revealing IP-Traffic Classification Rules. In *Proc. of IMC*.

[21] Fangfan Li, Abbas Razaghpanah, Arash Molavi Kakhki, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. 2017. liberate, (n): A Library for Exposing (Traffic-classification) Rules and Avoiding Them Efficiently. In *Proc. of IMC (IMC '17)*.

[22] Riccardo Ravaioli, Chadi Barakat, and Guillaume Urvoy-Keller. 2012. Chkdiff: checking traffic differentiation at internet access. In *Proc. of CoNEXT 2012 Student Workshop*. ACM.

[23] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. 2015. Towards a general solution for detecting traffic differentiation at the internet access. In *Proc. of the 23rd International Teletraffic Congress (ITC)*. IEEE.

[24] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. 2009. Detecting Network Neutrality Violations with Causal Inference. In *Proc. of ACM CoNEXT*.

[25] Huandong Wang, Fengli Xu, Yong Li, Pengyu Zhang, and Depeng Jin. 2015. Understanding mobile traffic patterns of large scale cellular towers in urban environment. In *Proc. of IMC*.

[26] Udi Weinsberg, Augustin Soule, and Laurent Massoulie. 2011. Inferring traffic shaping and policy parameters using end host measurements. In *Proc. INFOCOM*. IEEE.

[27] Ying Zhang, Z. Morley Mao, and Ming Zhang. 2009. Detecting Traffic Differentiation in Backbone ISPs with NetPolice. In *Proc. of IMC*.

APPENDIX

Appendices are supporting material that has not been peer reviewed.

A LAB FINDINGS WHILE VALIDATING OUR DIFFERENTIATION DETECTION

Here we expand upon findings specific to our lab tests performed in §5 to evaluate our differentiation detection scheme, presented in §4.

We observed 15 ISP-app pairs (all video streaming apps) affected by differentiation during our lab tests in 2018 and early Jan, 2019 (for these cases, Wehe detected throttling in all of the tests). The findings are consistent with (somewhat vague) ISP disclosures, such as "SD video streams up to 480p" for"all detectable video streaming."

In mid-January, we updated all recordings in Wehe, triggered by Amazon Prime Video's change from HTTPS to HTTP. We repeated our experiments and found identical results except for the case of Amazon and Vimeo. We found that Verizon and T-Mobile throttled both Amazon HTTP and HTTPS, but AT&T only throttled Amazon HTTP. The older recording of Vimeo traffic was throttled only by T-Mobile throughout our testing in January 2019, but the updated recording of Vimeo was not throttled by T-Mobile (or any other carrier). We believe this is because the DPI device in T-Mobile had not updated its rules to reflect Vimeo's change.

For every case of detected differentiation, we successfully identified the corresponding classification rules. For HTTPS traffic (i.e., YouTube, Netflix and Amazon) the classification rules are the same as reported by Li et al. [20, 21], i.e., the matching text is found in the SNI field of the TLS handshake, and consists of "googlevideo.com" for classifying YouTube, "nflxvideo.com" for Netflix, and ".aiv-cdn.net" for Amazon Prime. The matching content for Vimeo is "vimeo.akamaized.net", while the new Vimeo trace has SNI "vimeocdn.com" and not classified. For HTTP traffic (NBC Sports and Amazon Prime video), the classification rules partially check for a valid HTTP request (e.g., "GET" request with valid HTTP headers) and look for keywords in the payload. Example keywords include "akamaihd.net", "video" in "Content-Type" field. Table 5 presents detected DPI rules for HTTP traffic not reported in prior work.

B VIDEO RESOLUTION MAPPINGS

Table 6 shows the resolution mapping and the total pixel count of the various resolutions that were streamed on Amazon Prime Video, Netflix and YouTube during our video resolution experiments.

C DETAILS OF DETECTED THROTTLING

Table 7 complements the analysis in Table 4, providing additional details regarding the number of detected throttling cases per ISP-app pair, and the percent of total tests they represent for the ISP-app pair.

	NBC Sports	**Amazon Prime Video**
AT&T	"GET ", "video"	"GET ", "video"
T-Mobile	"GET ", "video", "nbcsports", "akamaihd.net"	"GET ", "video", "aiv-cdn.net"
Verizon	N/A	"GET ", "aiv-cdn.net"
Sprint	N/A	"GET ", "video", "aiv-cdn.net"

Table 5: Inferred DPI-classification keywords. N/A means we did not detect throttling, note that the "video" keyword appears in the HTTP *response* while the other keywords are in the HTTP *request*.

Labels	Resolutions	Total pixel count
LD	384x216	82,944
	426x240	102,240
	512x213	109,056
	480x270	129,600
	652x272	177,344
SD low	608x342	207,936
	710x296	210,160
	640x360	230,400
SD	640x480	307,200
	768x432	331,776
	720x480	345,600
	854x480	409,920
HD low	960x540	518,400
	1152x480	552,960
	1280x533	682,240
HD	1280x720	921,600

Table 6: Resolution mapping used in Figure 7.

		Rate	a		🦚		N		S		V		▶	
Country	ISP	(Mbps)	Tests	Perc.	Tests	Perc.	Tests	Perc.	Tests	Perc.	Tests	Perc.	Tests	Perc.
WiFi network														
Canada	Start	6	21	30%	11	20%								
UAE	Etisalat	0							23	100%				
US	Hughes	1					27	71%					15	35%
US	NextLink	4	5	36%	7	54%	14	88%			4	33%	16	94%
US	ViaNetTV	1					11	52%					14	58%
US	ViaSat	1					11	37%					20	24%
Cellular network														
Canada	Rogers	1.5					15	10%					34	12%
Canada	SaskTel	1.5					14	23%						
		3					15	25%						
Chile	Entel	1.5											16	53%
Germany	Telekom	1.5											16	15%
Israel	HOTmobile	1.5											16	70%
UAE	du	0							41	93%				
UAE	Etisalat	0							65	89%				
US	AT&T	1.5			6,581	71%	9,847	70%					16,774	74%
US	Boost	0.5							74	47%				
		2	73	40%			78	50%					131	44%
US	Cellcom	4	17	53%			16	67%					20	49%
US	Cricket	1.5	173	57%									660	72%
US	CSpire	1					8	62%					26	93%
US	FamilyMob.	1.5	24	89%	13	81%	19	83%					30	75%
US	GCI	1			9	41%	3	9%			3	14%	9	18%
		2	9	33%	5	23%	25	78%			8	36%	19	38%
US	iWireless	1.5					6	33%					8	29%
		3	7	44%	12	86%	9	50%					12	43%
US	MetroPCS	1.5	400	80%	317	86%	400	87%					636	79%
US	Sprint	1.5							905	12%				
		2	271	3%			432	5%					535	5%
US	Tracfone	2	49	35%									120	44%
US	T-Mobile	1.5	3,744	51%	3,381	64%	4,684	61%			162	5%	7,556	67%
US	Verizon	2	1,295	7%			1,758	9%					2,274	8%
		4	7,724	41%			8,831	43%					16,450	55%
US	Visible	2					18	78%					9	31%
US	XfinityMob.	2	24	62%			44	90%					34	79%
UK	giffgaff	1					21	100%					34	92%
UK	O2	1.5					49	64%					86	64%

Table 7: ISPs showing differentiation on particular apps, with the detected throttling rate in the second column. The data in each cell is the number of tests that detected differentiation along with the percentage of tests those represent of all tests for that app/ISP. Certain ISPs were observed with different throttling rates by different users, even for the same app.