BingeOn Under the Microscope:

Understanding T-Mobile's Zero-Rating Implementation

Arash Molavi Kakhki
Northeastern University
arash@ccs.neu.edu

Fangfan Li
Northeastern University
li.fa@husky.neu.edu

David Choffnes
Northeastern University
choffnes@ccs.neu.edu

Alan Mislove
Northeastern University
amislove@ccs.neu.edu

Ethan Katz-Bassett
University of Southern California
ethan.kb@usc.edu

1. INTRODUCTION AND BACKGROUND

The popularity of mobile devices for ubiquitous Internet access has led to exploding demand for relatively scarce cellular bandwidth. As a result, cellular operators increasingly use various techniques to manage their customers' demand on capacity, using traffic shaping, transcoding [9], and *zero-rating* [8]. With zero-rating, Internet Service Providers (ISPs) do not charge users for traffic sent to/from certain services, often because those services agree to use limited bandwidth resources.

Perhaps the most well-publicized case in the U.S. is T-Mobile's "*BingeOn*" service, which zero-rates video streams from a large number of partner sites. *BingeOn* has been highly controversial, due to concerns over network neutrality, user confusion, and technical downsides for users [10]. The resulting debate has led the EFF to call for T-Mobile to abandon *BingeOn*, and generated several responses from T-Mobile's CEO John Legere [1]. Importantly, there is little rigorous empirical data to inform the implications of zero-rating on network neutrality principles or customer-perceived Quality-of-Experience (QoE).

In this paper, we address this issue by conducting a study of T-Mobile's zero-rating policy and implementation to understand its implications for users and content providers in terms of data quota, performance, and QoE. We focus on T-Mobile and *BingeOn* due to their recent prominence, but we believe that lessons learned from this exercise will readily apply to other carriers using similar technologies to implement their policies.

We leverage our prior work on differentiation detection [9] to develop a suite of controlled tests. We deploy these tests to multiple *BingeOn* (and non-*BingeOn*) T-Mobile devices in the U.S., and correlate our tests with measurements of the billing records from these accounts. As a result, we are able to make significant headway in understanding *BingeOn*. Overall, we make five key contributions.

First, we characterize how *BingeOn* differentially impacts participating and non-participating providers. We determine that *BingeOn* is implemented solely by rate-limiting specific flows to 1.5 Mbps (consistent with the EFF's study). Despite the fact that T-Mobile claims that *BingeOn* provides "optimized streaming", there is no transcoding or optimization taking place.

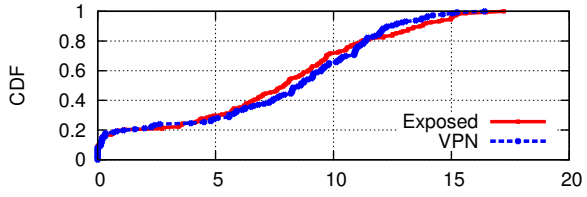
Second, we show how these differences translate to QoE, and how this QoE may impact users. We find that with *BingeOn* enabled, non-partner video flows see the same rate-limit (even though users are charged for these degraded flows). As a result, video services that do not support multiple quality levels can cause users to have poor video QoE due to high levels of buffering.

Third, we reverse-engineer the classifier used for enabling zero-rating, and show that *BingeOn* as implemented can have collateral damage by throttling flows that are not video content. We find that *BingeOn* is implemented using simplistic string matching, which can lead to false positives and negatives.

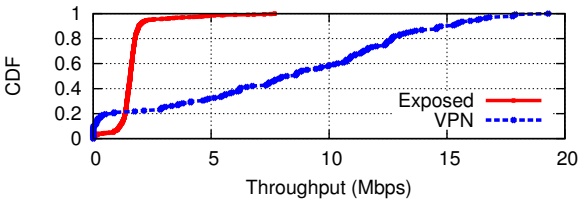
Fourth, we show that this implementation admits vulnerabilities such as free-riding. We demonstrate that the above implementation opens the door for abuse, as users can proxy their traffic with fake `Host` headers and receive zero-rating for arbitrary flows. We have responsibly disclosed this vulnerability to T-Mobile.

Fifth, we discuss how the above vulnerability is difficult to address using the current DPI-based approach, or alternative DPI approaches. This suggests the need for an alternative policy that can be completely and correctly implemented in existing middleboxes.

Taken together, our results indicate that the current *BingeOn* implementation can have negative impacts on both users and T-Mobile. If a user enrolls in *BingeOn*, all video traffic is rate-limited (even non-partners), and the rate-limiting can cause poor video quality for certain videos. Moreover, the implementation of *BingeOn* allows users to "steal" arbitrary amounts of data from T-Mobile. Last, our results serve to provide guidance



(a) *BingeOn* Disabled



(b) *BingeOn* Enabled

Figure 1: CDF of throughput for Netflix traffic replayed with *BingeOn* disabled and enabled. When *BingeOn* is off, no rate limit is imposed on Exposed or VPN-tunneled flows. With *BingeOn* enabled, VPN-tunneled traffic evades T-Mobile's detection and rate limits, while the Exposed replay is classified as Netflix and limited to 1.5Mbps.

for policy makers and regulators regarding zero-rating.

2. METHODOLOGY

There are two parts to our methodology: characterizing the behavior of *BingeOn* and reverse-engineering its implementation. Specifically, to characterize *BingeOn*, we need controlled experiments that reveal how *BingeOn* is impacting network performance and QoE. After confirming that *BingeOn* impacts our test traffic, we seek to reverse engineer its implementation to determine how T-Mobile determines that traffic should be classified as falling under *BingeOn* and whether this approach is susceptible to subversion.

2.1 Characterization

We begin by describing our approach to characterizing the behavior of *BingeOn*. We aim to address three questions with our methodology:

1. Does T-Mobile's BingeOn *give differential treatment to video traffic?* According to T-Mobile's *BingeOn* documentation, "*BingeOn* optimizes video quality for smartphone screens. It provides a great DVD-quality experience (typically 480p or better) for all detectable video, which can minimize buffering and maximize quality while using a fraction of the data" [5].

To determine whether T-Mobile indeed gives differential treatment to video traffic, we adopt the methodology used in our prior study of traffic differentiation [9]. To summarize, we record traffic traces from arbitrary mobile apps, then replay those recorded byte streams between a replay client and replay server under our control. As such, the only difference at the application layer between our recorded traces and replays is that our replay server usually does not have the same IP address as the server in the original recorded traces. As we demonstrated previously [9], not having the same IP address as the server does not affect DPI-based classification for all tested networks, including T-Mobile.

To determine if there is differentiation, we compare the throughput distribution for three types of replays: (1) the original packet payloads, (2) the original payloads encrypted via a VPN tunnel, and (3) the original payloads replaced with the same number of random bytes. If a DPI device is being used for classification, it should classify the exposed traffic (case 1) based on the application that generated it, and it should not be able to do so for encrypted or random bytes. To test whether this has an impact on performance, we conduct multiple rounds of tests in series and use statistical tests (a modified KS-test [9]) to identify differences in performance (i.e., throughput, latency) that indicate differentiation.

We conduct these tests separately with *BingeOn* enabled and disabled. We tested the apps listed in Table 3, which comprise popular streaming apps, some of which are participants in *BingeOn* [5].

2. Is BingeOn *traffic zero-rated?* Using the same methodology as described above, we determine whether the replayed traffic counts against the data quota for our testbed data plan. We do so using the self-service codes provided by T-Mobile for users to determine their data usage. Specifically, before we run each test, we determine the amount of data used, and we check again after the test. We observed the data usage counters to be updated in real-time, and we also confirmed this with T-Mobile's customer service.

3. What is the impact on QoE? To determine the impact of *BingeOn* on QoE, we use a popular video streaming service and gather commonly used QoE metrics such as video quality, quality changes, and rebuffering events. We conduct tests with *BingeOn* enabled and disabled, across several video bitrates.

2.2 Reverse Engineering

We investigated the *BingeOn* implementation with two central questions:

1. How does BingeOn *classify traffic?* Our prior differentiation work [9] suggested that DPI devices classify applications using regular expression matches on certain fields of HTTP requests and responses, and SNI fields in TLS handshakes. We thus use a modified version of the record/replay approach described in the previous section, where we modify various fields and determine their impact on *BingeOn* classification. We use two

	BingeOn disabled		*BingeOn* enabled	
	Exposed	Random	Exposed	Random
Throughput (Mbps)	7.707 (0.404)	7.231 (1.238)	**1.540 (0.003)**	7.205 (2.264)
RTT (ms)	206 (33)	201 (31)	**126 (2)**	141 (48)
Retransmission rate (%)	0.099 (0.093)	0.121 (0.1217)	**18.158 (0.732)**	0.272 (0.135)

Table 1: Netflix's average (std) throughput (a), RTT (b), and number of retransmissions (c) over T-Mobile's network with and without *BingeOn*. When *BingeOn* is enabled flows are consistently throttled to 1.5Mbps. When *BingeOn* is disabled, the average rate is higher and each replay varies based on available bandwidth and signal strength. The low RTT and high retransmission rates when *BingeOn* is enabled suggests that *BingeOn* uses policing for throttling.

sources of ground truth to validate whether *BingeOn* is being triggered on our test traffic: whether the traffic is zero-rated and/or it receives differentiated service.

2. Is BingeOn *susceptible to subversion?* Based on our observations from reverse engineering the classifier, we develop techniques to "trick" the *BingeOn* classifier into thinking that an application not participating in *BingeOn* is subject to *BingeOn*. For this, we develop a custom traffic-rewriting proxy and test the result by checking whether traffic is zero-rated and differentiated.

2.3 Testbed and Dataset

We conduct our experiments using Android phones with T-Mobile SIM cards. Each card is equipped with a data plan consisting of 6 GB of high-speed (4G) data. At any time, one of the SIM cards has *BingeOn* disabled, and the other one has it enabled. When running experiments, both SIM cards are in the same location, and their devices are connected to the same cell tower and have good reception. For tests using tethering, we used USB to tether a Mac laptop to the Android phones.

All of the results presented in this paper are based on tests conducted in Boston, MA. In total we ran more than 2000 traffic replays and more than 400 video streams against real apps using T-Mobile network, which add up to more than 20 GB of data. We conducted a small number of tests in Los Angeles, CA and found no difference in our results.

3. CHARACTERIZING BINGEON

In this section we quantify *BingeOn*'s impact on performance and video-streaming quality of experience.

3.1 Performance

Does *BingeOn* reduce video-streaming performance? To investigate this question, we begin using replays of Netflix traffic. Figure 1 shows individual examples of the impact of *BingeOn* on throughput. Each figure shows the cumulative distribution (CDF) of throughput samples for replays of Netflix traffic with *BingeOn* disabled (left) and enabled (right). The left figure shows that there are no significant visual differences between *VPN* and *Exposed* replays when *BingeOn* is disabled, but there is a clear difference when *BingeOn* is enabled in the bottom figure. We find that the average throughput for *Exposed* with *BingeOn* enabled is 1.5Mbps, about 80% lower than *Random* replays. Note that our results are consistent with EFF results [3].

We conducted 60 Netflix replays in the Boston area to build a large sample of data to characterize *BingeOn*. Table 1 shows the average and standard deviation of throughput, RTT, and retransmissions when replaying traffic with *BingeOn* enabled and disabled. We summarize our results below.

- *The* BingeOn *rate limit is ≈1.5 Mbps (Table 1).* This rate is sufficient for 480p videos on YouTube and "low quality" Netflix streaming, but significantly below requirements for the next-higher video quality categories (e.g., 720p for YouTube requires 2.5Mbps [2] and Netflix SD quality video requires 3Mbps [4]). Thus, T-Mobile's claim of supporting "480p or better" [5] video quality is misleading because we do not see evidence that they can support rates higher than 480p with *BingeOn* enabled.
- *The* BingeOn *infrastructure does not "optimize" video.* T-Mobile's CEO claimed that "*BingeOn* includes a proprietary technology to not only detect the video stream, but select the appropriate bit rate to optimize to the mobile device" [1]. Our differentiation detection methodology trivially reveals whether an ISP modifies content, e.g., for optimization. We found no modification to our replay content, nor any evidence that *BingeOn* behavior changed in response to the device we used (smartphone, or laptop).
- BingeOn *is implemented using policing.* There is low jitter and high retransmission rates when *BingeOn* is enabled (Table 1). This indicates a token bucket with a small (or no) queue which results in packets being dropped when there are no tokens available. [1]
- BingeOn*'s rate limit is cumulative for all flows from a single SIM.* If a customer streams simultaneous *BingeOn*-eligible videos (via tethering), the average throughput per stream will be lower than 1.5Mbps.

Is *BingeOn* traffic zero-rated while other video traffic is not? We address this question by investigat-

[1]Note that our replays do not adapt bitrates; rather, video traffic is replayed at the same bitrate it was recorded, which may be higher than supported by BingeOn.

Quality	Time to Start (sec)		video loaded in 60 secs (%)		#rebuffers		Buffer/Play Time (%)	
	BingeOn On	*BingeOn* Off	*BingeOn* On	*BingeOn* Off	*BingeOn* On	*BingeOn* Off	*BingeOn* On	*BingeOn* Off
auto*	1.54	1.32	0.04	0.03	0.0	0.0	2.48	2.15
hd1080	4.64	2.6	0.01	0.03	7.22	0.11	43.63	4.2
hd720	2.76	1.64	0.02	0.05	2.5	0.0	7.44	2.65
large	1.76	1.32	0.03	0.08	0.0	0.0	2.82	2.19
medium	1.58	1.08	0.04	0.12	0.0	0.0	2.56	1.76
small	1.26	1.0	0.07	0.15	0.0	0.0	2.07	1.63

Table 2: QoE metrics for a YouTube video in different qualities, averaged over 10 runs. *Auto selects hd1080 with *BingeOn* disabled and medium (360p) with *BingeOn* enabled.

ing data plan consumption before and after each test we run to tell if the traffic was zero-rated (no charge against the data plan). To confirm our methodology is valid, we contacted T-Mobile customer service, and they confirmed that "*BingeOn* services are whitelisted in their data usage counters," and that the counters were updated in real time.

We found that *BingeOn* traffic is typically zero-rated while traffic from services not participating in *BingeOn* was charged. When we tested with replays of video traffic from providers not participating in *BingeOn*, the data used was charged against our data plan. Thus, at the time of our experiments, YouTube traffic was throttled to 1.5 Mbps *and we were charged for the data*, whereas Netflix was similarly throttled and there was no data charged.[2] Such differential treatment of video services calls into question the policy's legality in the face of the FCC's Open Internet Order.

We found that BingeOn *behavior is not entirely consistent over time.* We encountered a small number of cases where a *BingeOn*-participating service's traffic was not zero-rated. These cases were transient, suggesting they are due to reasons such as buggy or overloaded infrastructure that supports *BingeOn*.

We conducted the same experiments for other *BingeOn* participants (e.g., HBOGo, ShowTime, and Hulu) and non-participants (e.g., Vimeo and Veoh), and found identical results. We also recorded and replayed other types of traffic (e.g., image download), and observed no rate limits imposed on them.

Does *BingeOn* work with tethering? T-Mobile states that *BingeOn* applies "when streaming video from one of *BingeOn* providers while tethering from a smartphone, tablet, or mobile internet device to a laptop, desktop, tablet, or handset" [5]. We investigate this by replaying *BingeOn* traffic on various devices while tethering from a smartphone. In February 2016, we found that *BingeOn* only works if tethering is done via USB. When tethering using the phone as a personal WiFi hotspot, *BingeOn*-eligible traffic was not rate limited or zero-rated. We tested this again in late March 2016, and *BingeOn* worked with tethering using both USB and a WiFi hotspot.

[2]Note that list of services participating in *BingeOn* changes over time (e.g., YouTube joined *BingeOn* just before submission of this paper), but we confirmed this behavior is true for every tested video service not participating in *BingeOn*.

3.2 Video Quality of Experience (QoE)

We now investigate how the policies revealed in the pervious section affects video-streaming QoE. We developed a tool that uses the YouTube iFrame Player API [7] to open an hour-long YouTube video, play it for 60 seconds and log the following QoE metrics: time to start the video, video quality, quality changes, rebuffering events, and fraction of video loaded. We use this tool to play YouTube videos with different bitrates, and with *BingeOn* enabled and disabled. We picked YouTube because it offers a large variety of video qualities, it supports HTTPS which prevents in-network caching, and it supports a rich API for gathering QoE metrics. While we necessarily focused on YouTube, we believe our results apply to other video services because the previous section showed *BingeOn* applies the same police to all video traffic.

Table 2 summarizes our findings.[3] As quality increases, QoE metrics degrade, leading to hd720 and higher qualities becoming unwatchable with *BingeOn* enabled. In contrast, the hd1080 quality can stream with good QoE when *BingeOn* is disabled.

BingeOn provides sufficient bandwidth for 480p and lower, but spends more time downloading video content, potentially leading to increased battery consumption due to preventing idle radio times. Interestingly, when *BingeOn* is enabled, YouTube selects medium (360p) quality, lower than the 480p promised by T-Mobile.

The lower bitrate occurs independent of the device screen size. For example, YouTube attempts to stream HD for tablets, but T-Mobile's throttling forces YouTube to adapt to lower qualities that result in visibly low resolution on large screens.

4. BINGEON UNDER THE MICROSCOPE

We now investigate how T-Mobile detects *BingeOn*-eligible traffic (§ 4.1) and how it can be exploited to zero-rate arbitrary traffic (§ 4.2).

4.1 Reverse Engineering Classification

T-Mobile claims that *BingeOn* includes a "proprietary technology" to detect video streams [1]. To un-

[3]We omitted 144p, hd1440, and hd2160 from the table as they follow the same trend.

Application	Detection criteria		How to evade detection?	
	BingeOn/Music Freedom*	Video	*BingeOn*	Video
Netflix	Specific GET arguments and the term "Netflix"	Same as *BingeOn*	Randomize GET argument and "Netflix" is reponse	Same as *BingeOn*
HBOGo	`Host` header ends with "hbogo.com"	`Content-Type` header (video/mp2t)	Randomize `Host` header	Randomize `Content-Type` header
ShowTime	`Host` header ends with "showvodhls.edgesuite.net**"	`Content-Type` header (video/mp2t)	Randomize `Host` header	Randomize `Content-Type` header
Hulu	`Host` header ends with "hulu.com"	`Content-Type` header (video/mp2t)	Randomize `Host` header	Randomize `Content-Type` header
Amazon Video	`Host` header ends with "amazonvod.loris.llnwd.net**"	`Content-Type` header (video/mp2t)	Randomize `Host` header	Randomize `Content-Type` header
Veoh	Not part of *BingeOn*	`Content-Type` header (video/mp4)	N/A	Randomize `Content-Type` header
Vimeo	Not part of *BingeOn*	Unknown***	N/A	N/A
Amazon Video	`Host` header ends with "amazonvod.loris.llnwd.net*"	`Content-Type` header (video/mp2t)	Randomize `Host` header	Randomize `Content-Type` header
YouTube (HTTP)	`Host` header ends with "googlevideo.com"	`Content-Type` header application/octet-stream	Randomize `Host` header	Randomize `Content-Type` header
YouTube (HTTPS)	Server name in the SNI ends with "googlevideo.com"	Same as *BingeOn*	Randomize "googlevideo" in the SNI	Same as *BingeOn*
Spotify*	"Spotify" in Host and User-agent headers	N/A	Randomize the term "Spotify"	N/A
Pandora*	`Host` header ends with "p-cdn.com"	N/A	Randomize `Host` header	N/A

Table 3: Summary of how each app is detected by T-Mobile, and what changes can evade detection. T-Mobile first checks for *BingeOn*. If it successfully matches a BingeOn app, the traffic will be zero-rated and throttled. Otherwise, it will next check for Video signatures and throttle such traffic (but not zero-rate) if it matches. ***Music Freedom*, is a program similar to *BingeOn*, which zero-rates music streaming apps [6]. **edgesuite.net (run by Akamai) and LLNWD.net are CDNs, serving ShowTime and Amazon videos. It is possible that these providers use other servers/CDNs too, hence different host names exist that result in zero-rating. ***We did not observe throttling, which we use to reverse engineer Video detection criteria. We suspect either that Vimeo has opted out of *BingeOn*, or *BingeOn* classifiers are not properly configured for Vimeo, which streams using HTTPS.**

derstand which features of the traffic are used to trigger detection, we use our testbed to replay a recorded trace multiple times, each time modifying different portions of traffic content (e.g., change the `Host` HTTP header) to observe its effect on classification. Our goal is to exhaustively understand how popular video services are detected, but we do not cover all video services.

To detect the impact of traffic contents on detection as *BingeOn*-eligible traffic, we enable *BingeOn* for a SIM card and log the data consumption and average throughput of each replay. Based on our analysis in the previous section, we detect the following policies:

- ***BingeOn*-eligible**: This applies to our replay if it zero-rated (data consumption does not increase) and its average throughput is 1.5Mbps.
- ***BingeOn*-ineligible video**: The replay is *not* zero-rated, but has an average throughput of 1.5Mbps.
- **Not video**: Traffic is not zero-rated or throttled.

Table 3 summarizes our findings for several popular video streaming services. One of the key take-aways is that *BingeOn* uses a DPI device that matches regular expressions to detect video and *BingeOn*-eligible traffic. As we discuss in the next section, this opens T-Mobile to exploitation for free-riding, and means that their policy can easily be erroneously applied to traffic. Our key findings are as follows:

- ***BingeOn* uses regular expressions to match on Host, Content-Type, and SNI fields.** The left two columns in Table 3 show that *BingeOn* is using simple regular-expression string matching based on pre-defined signatures. Classification first prioritizes the signature for *BingeOn* apps; if there is no app-specific match then *BingeOn* uses Content-Type signatures to detect non-*BingeOn* video streams.
- ***BingeOn*'s string-matching is brittle.** Once T-Mobile successfully matches a *BingeOn*-specific string in its classifier, it ignores all other fields that might support *or contradict* the classification. As we will show in § 4.2, this will allow exploitation of *BingeOn* to zero-rate arbitrary traffic.
- **IP addresses and port numbers have no impact on classification.** Our replay experiments use different IPs from the recorded services, and we also experimented with changing port numbers. We find that classification does not depend on the port, e.g., a request with Host:hbogo.com and video Content-Type to port 55555 (as opposed to the standard 80) on our replay server is still detected by T-Mobile as HBOGo.
- ***BingeOn* uses app-specific strings to throttle without zero-rating.** The "YouTube (HTTPS)" row in Table 3 shows that *BingeOn*-ineligible video

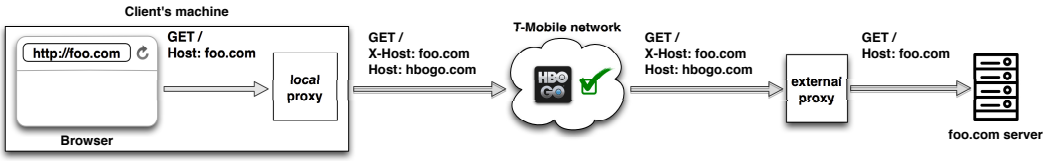


Figure 2: Subverting *BingeOn*: a) a local proxy copies the Host header into an optional parameter and overrides it with a *BingeOn*-enabled domain b) T-Mobile incorrectly classifies the traffic and zero-rates it c) an external proxy reverts the changes and forwards the request to destination.

streaming apps[4] are specifically targeted for throttling, in addition of general rules for all videos.

4.2 Subverting BingeOn

The previous section indicates that *BingeOn* traffic is classified strictly using simple text matching, which suggests that *we can exploit* BingeOn *to free-ride on T-Mobile* by modifying arbitrary HTTP traffic to masquerade as *BingeOn*-enabled (and thus zero-rated) activity. *We built a proxy that does this, and confirmed that it allows free-riding.* Our current implementation uses two proxies (Fig. 2): a local proxy on the user's device, and a proxy sitting outside T-Mobile's network (e.g., in a cloud data center).[5] The proxy works as follows. First, the local proxy stores the Host header in an `X-Host` header, then rewrites the Host header with a *BingeOn*-enabled host (e.g., `hbogo.com`) and forwards the request to a proxy located outside of T-Mobile's network. This causes T-Mobile's classifier to detect the traffic as *BingeOn*-enabled and zero-rate it. Next, the proxy outside of T-Mobile reverts the local proxy's changes and forwards the request to the final destination. Note that at the time of writing there is no need to modify any of the traffic in the reverse direction.

Free-riding on T-Mobile naturally raises ethical concerns. We privately notified T-Mobile of this vulnerability, and are currently discussing the issue with them.

5. DISCUSSION AND CONCLUSION

This paper provided a detailed look at zero-rating in a popular US carrier, both to characterize its impact on traffic as well as understand how the policy is implemented. We discovered a variety of behaviors that suggest T-Mobile is violating not only their own publicly stated policies but also the FCC's Open Internet Order. To summarize, *BingeOn* throttles all video traffic but charges for video from services not participating in *BingeOn*, there is no video- or screen-specific optimization, and this policy can have a negative impact on video-streaming QoE metrics.

Further, we identified how T-Mobile classified traffic for *BingeOn*, and found that its regular-expression-based approach is brittle, potentially inaccurate, and easily subverted to free-ride on T-Mobile. We have notified T-Mobile of these issues, but we believe that fixing such vulnerabilities is difficult to do using the current DPI-based approach. Namely, if T-Mobile changes their regular expressions to match the ones our proxy uses, we could easily use new ways of encoding our original header information. Further, if they use a static list of *BingeOn*-enabled apps' server IP addresses, they will not be able to easily account for dynamically redirected IPs commonly used by video-streaming CDNs. Last, more detailed traffic analysis will require more DPI resources, perhaps reducing the cost-effectiveness of any solution. We believe that the above cat-and-mouse game, if played out over time, will become disproportionately complex and expensive for T-Mobile. This issue arises because the simplistic DPI-based approach to traffic classification *does not match T-Mobile's* BingeOn *policy*. We believe the only way to prevent subversion is to specify a policy that can be *completely and correctly implemented* in existing middleboxes. Should there be no such solution, perhaps the best alternative would be to simply halt the current *BingeOn* policy.

[4]Before YouTube was added to *BingeOn*.

[5]Note that we would require only one local proxy if an HTTP server supports our subversion technique.

6. REFERENCES

[1] `https://twitter.com/JohnLegere/status/685201130427531264`.

[2] `https://support.google.com/youtube/answer/2853702`.

[3] EFF Confirms: T-Mobile's Binge On Optimization is Just Throttling, Applies Indiscriminately to All Video. `https://www.eff.org/deeplinks/2016/01/eff-confirms-t-mobiles-bingeon-optimization-just-throttling-applies`.

[4] Netflix internet connection speed recommendations. `https://help.netflix.com/en/node/306`.

[5] T-mobile bingeon. `http://www.t-mobile.com/offer/binge-on-streaming-video.html`.

[6] T-mobile music freedom. `http://www.t-mobile.com/offer/free-music-streaming.html`.

[7] Youtube iframe player api. `https://developers.google.com/youtube/iframe_api_reference`.

[8] Zero rating: What it is and why you should care. `https://www.eff.org/deeplinks/2016/02/zero-rating-what-it-is-why-you-should-care`.

[9] A. Kakhki, A. Razaghpanah, A. Li, H. Koo, R. Golani, D. Choffnes, P. Gill, and A. Mislove. Identifying traffic differentiation in mobile networks. IMC '15.

[10] B. van Schewick. T-mobile's binge on violates key net neutrality principles.