In-Band Quality Notification from End Users to ISPs

Anonymous Author(s)

Abstract

An ISP (Internet service provider) faces challenges in improving QoE (quality of experience) for end users, partly due to being just one element of the Internet's tiered connectivity structure that involves thousands of independent ISPs. This intricate structure complicates the out-of-band communication of QoE data from end users to the relevant ISPs. Additionally, the tussle between ISPs and OTT (over-the-top) providers, which employ end-to-end traffic encryption, makes OTT-ISP cooperation on QoE improvement unlikely in practice. To address these challenges, our paper proposes IQN (in-band quality notification), a new mechanism for in-band signaling of QoE information from the end user of an OTT service to pertinent ISPs. IQN operates without any support from the OTT provider. On the end-user device, IQN runs an end-user agent that leverages the OTT client's interface to estimate the end user's QoE and signal QoE estimates. IQN's ISP agent infers the QoE estimates from packet patterns observed at the ISP. We implement a YouStall prototype of IQN to detect and communicate video playback stalls of YouTube Live. Confirming the feasibility of IQN, our cloud-based evaluation shows that YouStall detects stalls at the end-user agent with nearly 100% precision and estimates the duration of significant stalls at the ISP agent with an average error less than 300 ms.

1 Introduction

Thousands of ISPs (Internet service providers) form a tiered structure to support global Internet connectivity for billions of end users. The complex structure weaves together ISPs of different kinds via diverse relationships. Figure 1 depicts ISP interconnections where a congested tier-2 ISP acts as a transit customer of a tier-1 ISP [18], buys partial transit [45] from another tier-1 ISP, peers with one tier-2 ISP via a private interconnection [20], purchases a remote-peering service [11] to reach an IXP (Internet exchange point) [2, 6], publicly peers at this IXP with another tier-2 ISP, and sells transit to two tier-3 ISPs. Most of the relationships in the Internet ecosystem are bilateral, and an end user typically does not know which other ISPs beyond the user's access ISP deliver the user's traffic [33]. The user has even worse visibility into which specific ISP is responsible for the network congestion impairing the user's QoE (quality of experience). Similarly, unless the affected end user is a direct customer, the congested ISP is generally unaware of either the identity or, due to the proliferation of NAT (network address translation) [26], the IP (Internet protocol) address of the user.

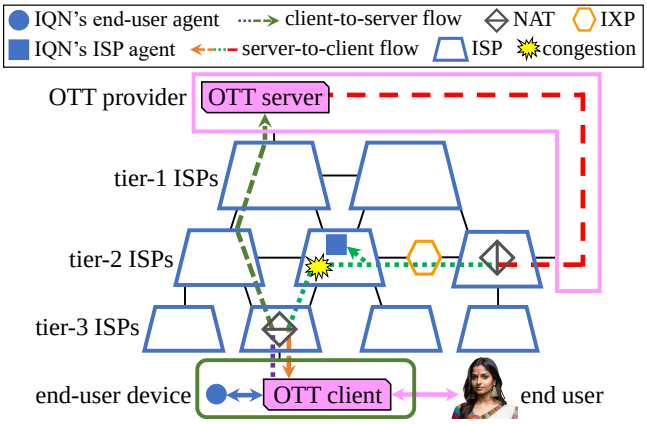


Figure 1: IQN's end-user and ISP agents in the Internet ecosystem with tiers of ISPs and an OTT client that offers an interactive interface on the end-user device.

ISPs strive to improve QoE for end users, whether they are direct customers or not, in order to maintain a good reputation and remain competitive in Internet connectivity markets. To resolve QoE impairments such as a stall of video playback [35], an ISP might subject a network flow to DPI (deep packet inspection) and QoE-aware traffic management, e.g., by means of rerouting or prioritization [7, 32]. End-user QoE improvement by ISPs faces challenges due to network neutrality regulations, such as the constraint of differentiating between applications during transient congestion only [40].

The ISPs' ability to enhance end-user QoE diminishes in their tussle [12] with OTT (over-the-top) providers, which include providers of content, applications, streaming and other services. OTT providers run their proprietary clients on end-user devices or utilize web browsers, e.g., by relying on JavaScript, to directly engage with the end users. To improve QoE, an OTT provider typically offers the end user an interface with interactive features, e.g., playback controls. OTT hypergiants, such as Google, Meta, and Netflix, leverage extensive cloud and edge infrastructures and operate their own private wide-area networks to deliver content from globally distributed locations [19, 46]. While exempt from network neutrality regulations [1], OTT providers cite fairness concerns to advocate for network neutrality restrictions on ISPs and prefer relegating ISPs to the role of dumb pipes for content delivery. End-to-end encryption, e.g., in the Google-designed QUIC (quick UDP Internet connections) protocol [27], is a part of this trend and hampers the ISPs'

ability to estimate end-user QoE via DPI. Alternative techniques for inferring QoE from monitored traffic, e.g., those based on packet sizes and timing [51], active collection of traffic fingerprints [56], and learning of growing sophistication [21, 30, 38, 39, 43, 48], tend to be less effective than DPI. Although OTT providers and ISPs share interests in end-user QoE improvement and have technical means for cooperation on this issue [15, 31, 55], an OTT provider typically goes it alone and limits its collaboration with ISPs to acquiring the needed Internet connectivity.

Whereas ISPs are on the defensive in their tussle with OTT providers, a potential strategy for ISPs is to obtain QoE feedback explicitly or implicitly from end users [42]. Measurements show that end users have an incentive to request ISPs to improve QoE for specific applications [52]. Existing mechanisms for direct communication of QoE information from an end user to an ISP include P4P (proactive network provider participation for P2P) [49] and ALTO (application-layer traffic optimization) [37]. However, this kind of mechanisms encounters practical complications because of its out-of-band nature. As Figure 1 illustrates, despite co-locating with many ISPs, even the OTT hypergiants are often two or more ISPs away from the end-user device [31]. Hence, the end user does not generally know which ISP to contact for QoE improvement. Additionally, as a network flow traverses the Internet, NAT alters the flow's 5-tuple description, which consists of the source and destination IP addresses and port numbers as well as the used protocol. Consequently, the end-user device and ISPs beyond a NAT device recognize the flow under different identifiers.

To tackle these challenges, our paper proposes IQN (in-band quality notification), a new practical mechanism for in-band QoE signaling from the end user of an OTT service to relevant ISPs. IQN operates without any support from the OTT provider. The key idea is to leverage the OTT client's interactive interface. IQN utilizes the interface to estimate the end user's QoE and signal the QoE estimates to the ISPs along the delivery path. As depicted in Figure 1, IQN relies on automated end-user and ISP agents, which run on the end-user device and in an ISP, respectively. To communicate a QoE estimate, the end-user agent leverages the OTT client's interface interactivity to programmatically issue a command that affects the pattern of packet transmission from the OTT server to the OTT client. Even though the OTT server encrypts the traffic, the ISP agent along the server-to-client path detects and interprets the changed packet pattern as an IQN signal.

The IQN mechanism is policy-free because it communicates QoE estimates without dictating a policy on usage of this information by ISPs. Each ISP independently decides how to react to the notification. Similar to an SOS signal, IQN acts as a distress signal sent by the end-user agent through the OTT service without knowing the intended ISP recipients of the signal. The end-user agent does not establish direct communication with any ISP and hopes that some ISP along the server-to-client path of the OTT service detects the IQN signal in monitored traffic, associates any reported QoE impairment with ongoing local congestion, and takes action to improve QoE for the end user. Furthermore, like the SOS signal, IQN is a signal intended for transient situations rather than persistent use, ensuring compliance with network neutrality regulations.

As a proof of concept for IQN, we prototype a YouStall system for estimation and in-band signaling of QoE in YouTube video streaming. YouStall focuses on playback stalls as one of the most prominent kinds of QoE impairments [35]. In YouStall, the end-user agent periodically captures screenshots of the YouTube client's playback area and detects a stall by recognizing a spinning icon. The objective of the stall detection is to avoid false positives, i.e., not to signal a stall spuriously. For in-band stall signaling, the end-user agent programmatically toggles the Autoplay switch in the YouTube client's interface, triggering a distinctive change in the packet transmission from the YouTube server to the YouTube client. By observing this characteristic packet pattern in a monitored network flow, the ISP agent infers a stall and estimates its duration. Our evaluation sets up an Amazon EC2 (elastic compute cloud) instance [36] to emulate an ISP and experiments with YouStall in the wild on YouTube Live [53] streams of the news, music, and sports genres.

Our experiments with YouStall confirm IQN's feasibility and effectiveness. The end-user agent detects stalls with precision close to 100%. Whereas the average stall duration across all three video genres exceeds 1.4 s, the ISP agent estimates the duration of significant stalls (those lasting at least 400 ms) with average MAE (mean absolute error) and RMSE (root mean square error) of 231 and 288 ms respectively. We also assess YouStall's parameter sensitivity and overhead. Overall, this paper makes the following main contributions:

(1) We propose IQN, a novel policy-free mechanism for in-band QoE signaling from the end user of an OTT service to the ISPs along the server-to-client path. IQN operates independently of any support from the OTT provider.
(2) The paper implements YouStall, an IQN prototype that enables ISPs to effectively detect stalls and estimate stall duration for YouTube Live streams.
(3) Our EC2-based evaluation of YouStall in the wild substantiates the feasibility and accuracy of the IQN mechanism.

Ethical considerations. The work does not involve human subjects and instead relies on an automated agent to programmatically interact with the OTT client's interface. Our experiments store neither content nor identifiers of streamers.

2 Motivation and principles

In this paper, we propose a new mechanism enabling an ISP to infer QoE delivered by an OTT service to end users. Our mechanism design follows a number of principles. As discussed in Section 1, the Internet ecosystem involves stakeholders with divergent interests and varying levels of influence in pursuing these interests. For example, OTT providers adopt end-to-end traffic encryption, thereby impeding the ability of ISPs to infer end-user QoE via DPI. Given the poor track record of cooperation between OTT providers and ISPs, a new mechanism for QoE inference by ISPs is unlikely to succeed if it requires any changes in the OTT service operation. Hence, our first design principle is:

PRINCIPLE 1 (OTT INDEPENDENCE). *The mechanism should operate without any support from the OTT provider.*

While ISPs face bleak chances of receiving assistance from OTT providers in end-user QoE inference, end users benefit from QoE improvement regardless of its source and have an incentive to cooperate with ISPs on this issue [52]. However, an OTT provider is generally reluctant to disclose detailed QoE information to its end users either, partly because such transparency might undermine the provider's business models. Instead, the OTT service offers limited cues, such as a spinning icon during a video stall, to communicate temporary QoE disruptions and thereby manage user expectations. If the new mechanism for QoE inference by an ISP involves the end user, the mechanism should constrain the involvement to the typical roles of casual and average users, i.e., to installing and running an application on the end-user device. Similarly, the designed mechanism should not overwhelm the ISP that infers QoE:

PRINCIPLE 2 (SIMPLICITY). *The mechanism should be simple and not overburden the involved ISPs and end users.*

Various practical aspects of the complex Internet ecosystem affect the feasibility and utility of QoE inference by ISPs. For instance, NAT and other factors introduce the possibility that a network flow alters its identity while traversing the Internet. Additionally, ISP-level routing in the tiered Internet structure is often asymmetric, and an OTT provider is capable of receiving requests through one ISP and responding through another. Hence, as Figure 1 illustrates, the client-to-server path might bypass the ISP responsible for QoE-impairing congestion. This implies the need for inferring QoE in ISPs that lie along the server-to-client path and are able to improve QoE, e.g., by prioritizing or rerouting QoE-impaired traffic. The designed mechanism should accommodate such practical considerations:

PRINCIPLE 3 (PRACTICALITY). *The mechanism should be practical to deploy in the current Internet.*

3 IQN design

Principle 3 in Section 2 requires consideration of current Internet realities, such as end-to-end traffic encryption by OTT services. Encryption poses serious challenges for inference accuracy when an ISP infers end-user QoE independently [51, 56]. Consequently, our approach to QoE inference provides ISPs with *assistance from the end user*.

Another practical constraint is the end user's potential unawareness of the intended assistance recipients due to multiple hops in the path from the OTT server to the end user [31], with the end user generally aware of only the last-hop ISP [33]. Furthermore, as a network flow traverses the Internet, the flow identity might change due to NAT [26], load balancing, and other commonly used techniques. Therefore, we avoid out-of-band mechanisms for direct information exchange between the end user and an ISP [37, 49] and instead proceed with *in-band notification* from the end user to the ISPs along the delivery path.

Principle 1 precludes any support from the OTT provider. Specifically, when the client-to-server path bypasses an ISP that lies along the server-to-client path, Principle 1 does not allow even minimal support of reflecting back any extra information, such as network cookies [52], received by the OTT server from the end-user device. Hence, our approach relies on the end user's *leverage of the OTT client's interactive interface* to induce the OTT server to transmit, in accordance with the OTT service's internal logic, a unique packet pattern to the OTT client. The ISPs along the server-to-client path detect and interpret this characteristic packet pattern as in-band notification of end-user QoE.

To summarize, the proposed IQN mechanism leverages the OTT client's interactive interface to assist ISPs along the server-to-client path via in-band notification of end-user QoE. In conformity with Principle 2, we design IQN as a simple *policy-free mechanism* that informs ISPs about end-user QoE without prescribing their responses. Akin to an SOS signal, IQN is a distress signal intended for transient QoE-impairing congestion, i.e., emergency situations rather than persistent usage, and issued along the server-to-client path to any ISP capable of resolving the issue. Each ISP independently decides how to react to the received IQN signal.

4 YouStall system

To understand the feasibility and utility of IQN, we prototype and experiment with the mechanism in YouStall, a system for detection and in-band user-to-ISP notification of YouTube's video stalls. YouStall consists of automated end-user and ISP agents running on end-user and ISP devices, respectively.

4.1 End-user agent

YouStall limits the role of the end user to installing and running a fully automated agent on the end-user device. While the device plays a YouTube video, the end-user agent operates in the background to execute the following two tasks: (1) stall detection and (2) in-band stall signaling.

Stall detection. The YouTube client's interface on the end-user device indicates a stall by showing an icon that spins in the center of the playback area. To monitor for stalls, the end-user agent captures a screenshot of the playback area at intervals of period p (every 200 ms by default). If the central part of two consecutive screenshots changes while the periphery remains the same, the end-user agent attributes the change to the spinning icon and infers a potential stall. The objective of the stall detection is to minimize false positives, i.e., avoid spurious notifications to ISPs about of a stall. Thus, the end-user agent incorporates additional logic to filter out its fictitious stall inferences caused by the end user's actions, such as moving the cursor or clicking on the progress bar. While the end-user agent does not detect stalls shorter than p, this feature aligns well with YouStall's intention to inform ISPs about longer stalls creating noticeable disruption in end-user QoE and resolvable by the ISPs.

In-band stall signaling. Upon each detection of a stall, the end-user agent sends an IQN signal by utilizing again the YouTube client's interface. Whereas we develop YouStall to illustrate IQN feasibility without attempting to make the prototype perfect, our choice of a specific signaling technique seeks to avoid disrupting the user's experience: the end-user agent emits the IQN signal by programmatically toggling the Autoplay switch an even number of times in rapid succession, with intervals shorter than 5 ms between consecutive toggles. The entire series of n toggles occurs too quickly to be perceived by the human eye and concludes with the cursor and Autoplay switch in their original positions. By default, we set n to 4 as the smallest even number that triggers an easily recognizable pattern in the server-to-client transmission.

For each toggle in the IQN signal, the YouTube client transmits an HTTP/3 (hypertext transfer protocol version 3) POST request to the YouTube server, which then updates its Autoplay settings and sends an HTTP/3 response with the "200 OK" status over QUIC to the YouTube client. Although a monitored network flow might include other QUIC packets that contain "200 OK" confirmations, a quick series of such packets is a rare pattern. YouStall utilizes this distinctive pattern as a basis for stall inference by the ISP agent.

4.2 ISP agent

The ISP agent faces several challenges in inferring stalls from IQN signals. Since YouTube encrypts its traffic and hides "200 OK" confirmations in QUIC packets, the ISP agent analyzes packet sizes to identify QUIC packets associated with IQN signaling, which range from 1,180 to 1,288 bytes. However, the monitored network flow occasionally contains unrelated packets within this size range, requiring the ISP agent to discern which packets correspond to in-band stall signals. Additionally, even when the end-user agent issues a series of n-toggle IQN signals at intervals of period p to notify about a long stall, the path from the YouTube client through the YouTube server to the ISP agent might introduce significant jitter [8] due to end-host or in-network processing, reorder or lose related packets, and substantially distort the packet sequence arriving to the ISP agent.

To address the above challenges, the ISP agent monitors candidate packets, i.e., those within the targeted size range, of each network flow separately. The inference algorithm operates in either *out-of-stall* or *in-stall* mode. In the out-of-stall mode, the ISP agent slides a *stall-start window* over the arrival times and infers the start of a new stall if the window covers at least n packet arrivals. The algorithm then records the first of these arrival times as start time s of the stall and switches to the in-stall mode. In the in-stall mode, the ISP agent slides a *stall-end window* over the arrival times and infers the end of the current stall if this window covers less than n packet arrivals. The algorithm then records the first of these arrival times as end time e of the stall, estimates the stall duration as $e - s$, and switches to the out-of-stall mode.

We keep the algorithm simple by sizing both stall-start and stall-end windows in relation to period p instead of introducing new independent parameters. Since the end-user agent generates IQN signals to avoid false positives, we size the stall-start window to $w_s = \frac{3p}{2}$, i.e., 50% longer than signaling period p. Despite potential timing distortions by the delivery path, we expect this duration to be long enough for the arrival of at least n packets corresponding to the initial in-band signaling of a stall. On the other hand, $w_s = \frac{3p}{2}$ is short enough to avoid inferring the start of a stall spuriously in response to the arrival of unrelated candidate packets. We empirically size the stall-end window to $w_e = 4p$ to mitigate false negatives, which might occur when the end-user agent skips an IQN signal during an actual stall. Because unrelated candidate packets are infrequent, they do not disrupt inferring the end of a stall when w_e is as long as $4p$. Additionally, $w_e = 4p$ is shorter than the typical gaps between actual stalls, thus avoiding an inference of multiple stalls as one.

4.3 Implementation

We implement YouStall in Python 3.10. YouStall's end-user agent utilizes Pillow [13] for image manipulation and sets the PAUSE parameter of PyAutoGUI [41] to 0 for rapid toggling of the Autoplay switch. The ISP agent employs pyshark [24] for real-time packet capture and analysis. At the time of paper

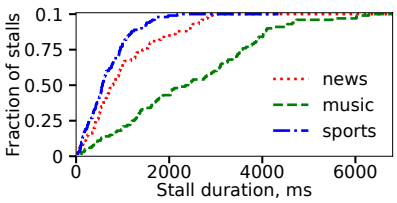


Figure 2: Ground-truth stall duration.

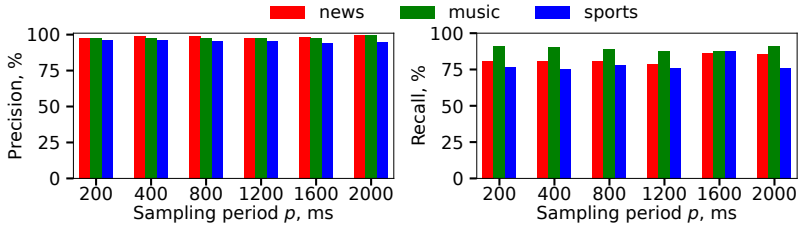
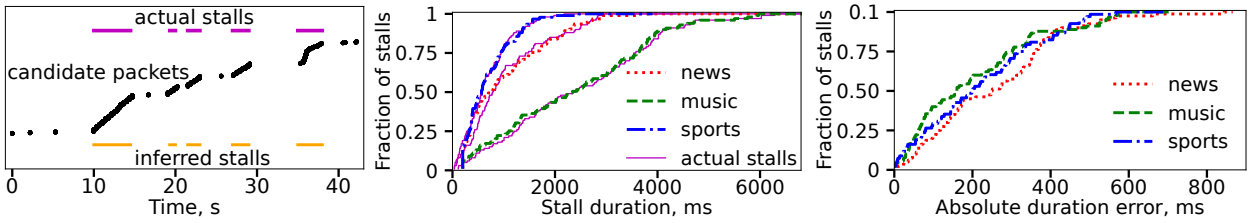


Figure 3: Precision and recall of stall detection by the end-user agent.



(a) An operational example **(b) Duration of inferred vs. actual stalls** **(c) Duration error for significant stalls**

Figure 4: Inference of stalls and their duration by YouStall's ISP agent.

publication, we openly release the YouStall code and our experimental configurations.

5 Evaluation

5.1 Experimental setup

To evaluate YouStall, we utilize an Intel i7 machine (six cores, 2.6-GHz CPUs, 16-GB RAM, Linux Ubuntu 22.04.4 LTS) as the end-user device. The ISP agent runs on an EC2 instance (one-core 2.4-GHz Intel Xeon CPU, 1-GB RAM, Linux Ubuntu 24.04.4 LTS) located about 1,000 miles away. We tunnel Internet traffic of the end-user device to the EC2 instance with OpenVPN [23]. An automated script plays YouTube Live [53] video streams from three genres—news, music, and sports—on the end-user device via the Google Chrome browser.

Since YouTube stalls are typically infrequent, our evaluation induces a significant number of stalls—100 per genre—by fixing the video resolution at 720p and using tcconfig [22] to limit the YouTube traffic to 1 Mbps [28]. We collect ground truth on the stalls by employing Selenium [16] and, in particular, tracking the ytp-spinner tag that renders the spinning icon in YouTube's client interface. We synchronize the timelines of the stalls and captured screenshots. To assess YouStall's memory [47] and CPU (central processing unit) overhead, we utilize psutil [34].

5.2 Experimental results

Figure 2 depicts the distributions of the stall duration for the three genres according to the ground truth. The news, music, and sports genres exhibit distinct stalling profiles, with the average stall duration of 1.1, 2.6, and 0.7 s, respectively.

Figure 3 presents the precision and recall of stall detection by YouStall's end-user agent according to the synchronized timelines of actual stalls and captured screenshots. The sampling period has minimal impact on these metrics. The end-user agent detects stalls with nearly 100% precision and largely avoids signaling a stall spuriously. However, the recall is lower because, by design, the end-user agent detects a stall only upon capturing the second screenshot during the stall, implying that YouStall does not detect stalls of a shorter duration than p. Because shortening the value of p increases the number of detected stalls, we set the default sampling period to the shortest examined value, i.e., $p = 200$ ms.

Figure 4a illustrates the operation of YouStall's ISP agent. The horizontal magenta segments indicate actual stalls, which prompt the end-user agent to emit IQN signals towards the ISP agent. Each black dot represents the arrival of a candidate packet to the ISP agent. Between stalls, the ISP agent observes infrequent arrivals of candidate packets unrelated to IQN signaling. During a stall, the ISP agent detects a higher arrival rate of candidate packets and infers the stall correctly. Figure 4a depicts the inferred stalls as horizontal orange segments.

Figure 4b plots the distributions of the inferred stall duration for the news, music, and sports genres and employs three additional solid magenta lines for depicting the respective distributions of the actual stall duration. While the distributions of the actual stall duration for the three genres are distinct, the distributions of the inferred and actual stall duration for each

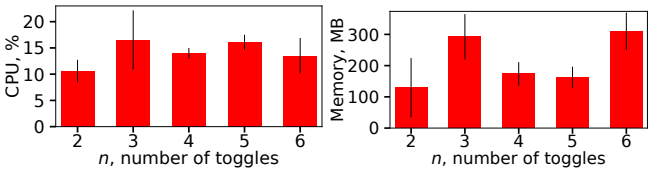


Figure 5: Overhead of YouStall's end-user agent.

particular genre are very similar, suggesting that YouStall infers stall-duration distributions accurately.

Because we experiment with YouStall in the wild with substantial jitter (up to 1 s) along the path from the end-user device to the ISP agent, evaluation of the inference accuracy for specific stalls is challenging. We utilize the end-user and ISP-agent timelines to match the actual and inferred stalls (within a window of 1 s) and, for significant actual stalls lasting at least 400 ms, confidently determine 238 one-to-one mappings between inferred and actual stalls across the three genres (i.e., for 238 out of the total 300 actual stalls). The average MAE and RMSE over these 238 confident one-to-one mappings of inferred and actual significant stalls are 231 and 288 ms, respectively, with Figure 4c plotting the absolute error of the inferred stall duration separately for each genre. The results indicate that YouStall estimates the duration of significant stalls with relatively low errors.

Figure 5 reports on overhead of YouStall's end-user agent. The results are from four runs with 10 actual stalls. The CPU and memory consumption is around 12% and 200 MB, respectively. The overhead is affordable and does not strongly depend on the number of toggles constituting an IQN signal.

6 Related work

QoE inference from encrypted traffic. [51] considers packet sizes and timing to identify video segments in encrypted traffic. [56] relies on combinatorial matching to detect the video resolution when QUIC multiplexes video and audio content. [30], [21], and [39] infer various QoE metrics by applying decision trees, random forests, and convolutional neural networks, respectively. Other solutions for QoE detection from encrypted traffic include [29], [38], [43], and [48]. Despite the increasing power of the applied learning techniques, independent inference of QoE by an ISP struggles to achieve high performance. In contrast, IQN provides assistance from the end user for enabling an ISP to infer QoE accurately.

OTT-ISP cooperation. Collaboration with OTT providers and CDNs (content delivery networks) [25, 57], which help the OTT providers enhance end-user QoE, promises tangible benefits for an ISP. Prior work on OTT-CDN collaboration deals with CDN-server placement and end-user assignment [17], reconfigurable topologies [54], privacy preservation [3], and temporary increases of the network bandwidth allocation [44]. [15] compares three techno-economic models for OTT-ISP cooperation. [55] leverages software-defined networking to jointly optimize intra-domain and inter-domain routing for an ISP in collaboration with OTT providers. [31] enables cooperation between OTT hypergiants and remote ISPs without establishment of direct peering. In spite of the available technical means, OTT providers are reluctant to cooperate with ISPs on end-user QoE inference in practice. We design IQN to operate without any support from OTT providers.

QoE signaling. CMCD (common media client data) [10] supports notification from OTT providers' proprietary video players to CDNs and is not directly accessible to end users. While P4P [49] and ALTO [37] enable the end user to report QoE directly to an ISP, these out-of-band mechanisms have limited practical utility due to such issues as NAT. Network cookies [52] constitute an in-band notification mechanism but, unlike IQN, require support from the OTT server.

Leverage of client interfaces. VideoEye [50] records the video player's screen for offline inference of QoE from the recorded video. Tero [4, 5] extracts QoE from thumbnails in gaming footage. Unlike VideoEye and Tero, the YouStall prototype of IQN captures screenshots for real-time detection and in-band signalling of QoE to ISPs. While [9] evaluates how various interactions with the video player's interface [14] affect QoE of encrypted video sessions, YouStall uses a particular interaction with the client's interface to induce a distinctive pattern of packet transmission from the YouTube server.

7 Discussion and conclusion

Whereas IQN operates without support from OTT providers, the latter can block or disrupt IQN signals sent from end users to ISPs. Such aggressive actions might prompt the implementation of neutrality regulations for OTT providers, akin to the current rules governing ISPs.

Our YouStall prototype relies on a simple yet effective algorithm for inferring QoE from IQN signals. Application of more advanced techniques, such as deep learning, has a potential to further improve QoE inference.

While this paper explores IQN as a novel mechanism for in-band signaling to ISPs about QoE impairment, the notified ISPs encounter extra challenges in resolving the QoE impairment, e.g., what actions to apply to which network flows. Closing the loop to enhance end-user QoE represents a natural extension of our work.

Overall, our preliminary study confirms the feasibility of the IQN mechanism and opens opportunities for future work on its various aspects. These research directions include selecting the OTT client's interface feature for IQN signaling, encoding of end-user QoE information into IQN signals, and integrating the new IQN approach with traditional QoE inference techniques.

References

[1] Muhammad Abdullah, Pavlos Nikolopoulos, and Katerina Argyraki. 2023. Caching and Neutrality. In *HotNets 2023*.

[2] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. 2012. Anatomy of a Large European IXP. In *SIGCOMM 2012*.

[3] Kutalmiş Akpinar and Kien A. Hua. 2020. PPNet: Privacy Protected CDN-ISP Collaboration for QoS-Aware Multi-CDN Adaptive Video Streaming. *ACM TOMM* 16, 2 (2020), 1–23.

[4] Catalina Alvarez and Katerina Argyraki. 2023. Learning a QoE Metric from Social Media and Gaming Footage. In *HotNets 2023*.

[5] Catalina Alvarez and Katerina Argyraki. 2023. Using Gaming Footage as a Source of Internet Latency Information. In *IMC 2023*.

[6] Gianni Antichi, Ignacio Castro, Marco Chiesa, Eder L. Fernandes, Remy Lapeyrade, Daniel Kopp, Jong Hun Han, Marc Bruyere, Christoph Dietzel, Mitchell Gusat, Andrew W. Moore, Philippe Owezarski, Steve Uhlig, and Marco Canini. 2017. ENDEAVOUR: A Scalable SDN Architecture for Real-World IXPs. *IEEE JSAC* 35, 11 (2017), 2553–2562.

[7] Maria Apostolaki, Ankit Singla, and Laurent Vanbever. 2021. Performance-Driven Internet Path Selection. In *SOSR 2021*.

[8] Venkat Arun, Mohammad Alizadeh, and Hari Balakrishnan. 2022. Starvation in End-to-End Congestion Control. In *SIGCOMM 2022*.

[9] Ivan Bartolec. 2021. Performance Estimation of Encrypted Video Streaming in Light of End-User Playback-Related Interactions. In *MMSys 2021*.

[10] Abdelhak Bentaleb, May Lim, Mehmet N. Akcay, Ali C. Begen, and Roger Zimmermann. 2021. Common Media Client Data (CMCD): Initial Findings. In *NOSSDAV 2021*.

[11] Ignacio Castro, Juan Camilo Cardona, Sergey Gorinsky, and Pierre Francois. 2014. Remote Peering: More Peering without Internet Flattening. In *CoNEXT 2014*.

[12] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. 2005. Tussle in Cyberspace: Defining Tomorrow's Internet. *IEEE/ACM ToN* 13, 3 (2005), 462–475.

[13] Jeffrey A. Clark. 2024. pillow 10.3.0. Python Sofware Foundation. https://pypi.org/project/pillow/.

[14] Cristiano P. Costa, Italo S. Cunha, Alex Borges, Claudiney V. Ramos, Marcus M. Rocha, Jussara M. Almeida, and Berthier Ribeiro-Neto. 2004. Analyzing Client Interactivity in Streaming Media. In *WWW 2004*.

[15] Alessandro Floris, Arslan Ahmad, and Luigi Atzori. 2018. QoE-Aware OTT-ISP Collaboration in Service Management: Architecture and Approaches. *ACM TOMM* 14, 2s (2018), 1–24.

[16] Python Software Foundation. 2024. selenium 4.22.0. https://pypi.org/project/selenium/.

[17] Benjamin Frank, Ingmar Poese, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. 2013. Pushing CDN-ISP Collaboration to the Limit. *SIGCOMM CCR* 43, 3 (2013), 34–44.

[18] Lixin Gao. 2001. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM ToN* 9, 6 (2001), 733–745.

[19] Petros Gigis, Matt Calder, Lefteris Manassakis, George Nomikos, Vasileios Kotronis, Xenofontas Dimitropoulos, Ethan Katz-Bassett, and Georgios Smaragdakis. 2021. Seven Years in the Life of Hypergiants' Off-Nets. In *SIGCOMM 2021*.

[20] Vasileios Giotsas, Georgios Smaragdakis, Bradley Huffaker, Matthew Luckie, and kc claffy. 2015. Mapping Peering Interconnections to a Facility. In *CoNEXT 2015*.

[21] Craig Gutterman, Katherine Guo, Sarthak Arora, Trey Gilliland, Xiaoyang Wang, Les Wu, Ethan Katz-Bassett, and Gil Zussman. 2020. Requet: Real-Time QoE Metric Detection for Encrypted YouTube Traffic. *ACM TOMM* 16, 2s (2020), 1–28.

[22] Tsuyoshi Hombashi. 2024. tcconfig 0.7.0a1. Python Sofware Foundation. https://pypi.org/project/tcconfig/0.7.0a1/.

[23] OpenVPN Inc. 2024. OpenVPN. https://openvpn.net/.

[24] KimiNewt. 2024. pyshark 0.6. Python Software Foundation. https://pypi.org/project/pyshark/.

[25] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. 2009. Moving Beyond End-to-End Path Information to Optimize CDN Performance. In *IMC 2009*.

[26] Chang Lan, Justine Sherry, Raluca Ada Popa, Sylvia Ratnasamy, and Zhi Liu. 2016. Embark: Securely Outsourcing Middleboxes to the Cloud. In *NSDI 2016*.

[27] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *SIGCOMM 2017*.

[28] Jiamo Liu, David Lerner, Jae Chung, Udit Paul, Arpit Gupta, and Elizabeth Belding. 2024. Watching Stars in Pixels: The Interplay of Traffic Shaping and YouTube Streaming QoE over GEO Satellite Networks. In *PAM 2024*.

[29] Tarun Mangla, Ellen Zegura, Mostafa Ammar, Emir Halepovic, Kyung-Wook Hwang, Rittwik Jana, and Marco Platania. 2018. VideoNOC: Assessing Video QoE for Network Operators Using Passive Measurements. In *MMSys 2018*.

[30] M. Hammad Mazhar and Zubair Shafiq. 2018. Real-Time Video Quality of Experience Monitoring for HTTPS and QUIC. In *INFOCOM 2018*.

[31] Cristian Munteanu, Oliver Gasser, Ingmar Poese, Georgios Smaragdakis, and Anja Feldmann. 2023. Enabling Multi-Hop ISP-Hypergiant Collaboration. In *ANRW 2023*.

[32] Ashkan Nikravesh, David Ke Hong, Qi Alfred Chen, Harsha V. Madhyastha, and Z. Morley Mao. 2016. QoE Inference Without Application Control. In *Internet-QoE 2016*.

[33] Larry L. Peterson and Bruce S. Davie. 2021. *Computer Networks: A Systems Approach*. Morgan Kaufmann.

[34] Giampaolo Rodola. 2024. psutil 6.0.0. Python Software Foundation. https://pypi.org/project/psutil/.

[35] Raimund Schatz, Tobias Hoßfeld, and Pedro Casas. 2012. Passive YouTube QoE Monitoring for ISPs. In *IMIS 2012*.

[36] Amazon Web Services. 2024. Amazon EC2. https://aws.amazon.com/ec2.

[37] Stanislav Shalunov, Roome Wendy, Richard Woundy, Stefano Previdi, Sebastian Kiesel, Richard Alimi, Reinaldo Penno, and Y. Richard Yang. 2014. Application-Layer Traffic Optimization (ALTO) Protocol. IETF RFC 7285.

[38] Taveesh Sharma, Tarun Mangla, Arpit Gupta, Junchen Jiang, and Nick Feamster. 2013. Estimating WebRTC Video QoE Metrics Without Using Application Headers. In *IMC 2023*.

[39] Meng Shen, Jinpeng Zhang, Ke Xu, Liehuang Zhu, Jiangchuan Liu, and Xiaojiang Du. 2020. DeepQoE: Real-Time Measurement of Video QoE from Encrypted Traffic with Deep Learning. In *IWQoS 2020*.

[40] Volker Stocker, Georgios Smaragdakis, and William Lehr. 2020. The State of Network Neutrality Regulation. *SIGCOMM CCR* 50, 1 (2020), 45–59.

[41] Al Sweigart. 2024. PyAutoGUI 0.9.54. Python Software Foundation.https://pypi.org/project/PyAutoGUI/.

[42] Aryan Taneja, Rahul Bothra, Debopam Bhattacherjee, Rohan Gandhi, Venkata N. Padmanabhan, Ranjita Bhagwan, Nagarajan Natarajan,

Saikat Guha, and Ross Cutler. 2023. Don't Forget the User: It's Time to Rethink Network Measurements. In *HotNets 2023*.

[43] Tisa-Selma, Abdelhak Bentaleb, and Saad Harous. 2020. Inferring Quality of Experience for Adaptive Video Streaming over HTTPS and QUIC. In *IWCMC 2020*.

[44] Tuan Tran, Dylan Gageot, Christoph Neumann, Guillaume Bichot, Abderrahmen Tlili, Karim Boutiba, and Adlen Ksentini. 2024. On the Benefits and Caveats of Exploiting Quality on Demand Network APIs for Video Streaming. In *NOSSDAV 2024*.

[45] Vytautas Valancius, Cristian Lumezanu, Nick Feamster, Ramesh Johari, and Vijay Vazirani. 2011. How Many Tiers? Pricing in the Internet Transit Market. In *SIGCOMM 2011*.

[46] Kevin Vermeulen, Loqman Salamatian, Sang Hoon Kim, Matt Calder, and Ethan Katz-Bassett. 2023. The Central Problem with Distributed Content: Common CDN Deployments Centralize Traffic In A Risky Way. In *HotNets 2023*.

[47] Talha Waheed, Ihsan Ayyub Qazi, Zahaib Akhtar, and Zafar Ayyub Qazi. 2022. Coal Not Diamonds: How Memory Pressure Falters Mobile Video QoE. In *CoNEXT 2022*.

[48] Sarah Wassermann, Michael Seufert, Pedro Casas, Li Gang, and Kuang Li. 2020. ViCrypt to the Rescue: Real-Time, Machine-Learning-Driven Video-QoE Monitoring for Encrypted Streaming Traffic. *IEEE TNSM* 17, 4 (2020), 2007–2023.

[49] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. 2008. P4P: Provider Portal for Applications. In *SIGCOMM 2008*.

[50] Shichang Xu, Eric Petajan, Subhabrata Sen, and Z. Morley Mao. 2020. What You See Is What You Get: Measure ABR Video Streaming QoE via On-Device Screen Recording. In *NOSSDAV 2020*.

[51] Shichang Xu, Subhabrata Sen, and Z. Morley Mao. 2020. CSI: Inferring Mobile ABR Video Adaptation Behavior Under HTTPS and QUIC. In *EuroSys 2020*.

[52] Yiannis Yiakoumis, Sachin Katti, and Nick McKeown. 2016. Neutral Net Neutrality. In *SIGCOMM 2016*.

[53] Google LLC YouTube. 2024. YouTube Live. https://www.youtube.com/live.

[54] Johannes Zerwas, Ingmar Poese, Stefan Schmid, and Andreas Blenk. 2022. On the Benefits of Joint Optimization of Reconfigurable CDN-ISP Infrastructure. *IEEE TNSM* 19, 1 (2022), 158–173.

[55] Yimeng Zhao, Ahmed Saeed, Mostafa H. Ammar, and Ellen W. Zegura. 2019. Unison: Enabling Content Provider/ISP Collaboration Using a vSwitch Abstraction. In *ICNP 2019*.

[56] Yuanjie Zhao, Hua Wu, Liujinhan Chen, Songtao Liu, Guang Cheng, and Xiaoyan Hu. 2024. Identifying Video Resolution from Encrypted QUIC Streams in Segment-Combined Transmission Scenarios. In *NOSSDAV 2024*.

[57] Jiangchen Zhu, Kevin Vermeulen, Italo Cunha, Ethan Katz-Bassett, and Matt Calder. 2022. The Best of Both Worlds: High Availability CDN Routing without Compromising Control. In *IMC 2022*.