Application-Aware Analysis of Network Neutrality: A Scalable Real-Time Method

Péter Orosz, Tamás Skopkó, Tamás Marosits

Abstract—**Internet access subscribers expect a satisfying quality of experience for any accessed service, independently from time, place, and service- and content-type. Besides the ever-increasing amount of Internet data, the spectrum of video service platforms offering sharing and streaming also got significantly more comprehensive. Internet access providers try to avoid the exhaustion of network bandwidth by investing in network capacity or setting up higher-level resource management within their infrastructure. The primary question in this domain is how resource management constrains the subscriber to access an arbitrary service and experience good service quality. This question directly relates to network neutrality fundamentals.**

This paper presents a real-time full-reference objective method to assess network neutrality. It contributes three novelties to support user-centric analysis of potential restraints affecting Internet access quality: i) the proposal supports application-specific measurements and involves real content and real traffic, ii) the measured traffic originates from the content provider's cloud infrastructure, iii) reference is created in real time. Accordingly, the proposal introduces a novel measurement layout. The key component is the emulated client that provides the real-time reference by emulating the access properties of the real client and accessing the same content simultaneously.

We demonstrate the method's feasibility with an application-aware proof-of-concept use case: video streaming from a public VoD provider. We have validated the method against the emulated network parameters using an extensive series of laboratory measurements.

Index Terms—**Network neutrality, quality measurement, video streaming, objective quality model.**

I. INTRODUCTION

CLOUD-BASED services have become the dominators of global Internet communication in the last decade. Evolved data-center technologies and architectures opened the way toward centralized, global service platforms. From an Internet provider's perspective, it is a crucial challenge to manage network resources optimizing its subscribers' quality of experience for those cloud services that dominate their traffic mix. Its endeavor to control resources, i.e., traffic engineering based on service popularity, may offend the original best-effort paradigm of the global Internet. Handling a wide scale of traffic types requires tools and techniques that support the network to fulfill various transmission requirements. Since the best-effort communication model effectively fosters new services and technologies, prioritizing the service platforms based on popularity ranking may set up numerous constraints to new technologies and services. The primary question in this domain is how advanced resource management constraints the subscriber to access an arbitrary service and experience good service quality. This question directly relates to network neutrality fundamentals which are supported by the legal framework Regulation (EU) 2015/2120 of the European Parliament and the Council [1]. The regulation and the status quo imply a pivotal question: How can consumers and authorities objectively verify Internet access neutrality for a wide range of services in the cloud-era?

Although the regulation itself is intended to be the legal mechanism in this field, a complete consumer protection suit also requires a dedicated technological background to enable verifying the neutrality of an Internet access, and the latter is still missing. The primary reason for this large gap between the availability of legal and technical tools is that the scientific fundamentals providing the underlying measurement paradigm and scalable methods are yet to be established. Analyzing a wide scale of network neutrality attributes (defined by EU's BEREC Office [2] [3] and introduces new ones for cloud services) is a research "green-field" without dedicated standards and methods.

The network neutrality paradigm expects Internet access providers to treat all user traffic equally, independently from the type of device, platform, service, and content [1]. In contrast, the last decade has seen many user restrictions and traffic differentiation cases that violate the neutrality principle. Early cases typically covered service functionality restrictions by explicitly blocking domains, IP address ranges, and transport protocol port identifiers. With the evolution of deep packet inspection (DPI) technologies, operators can identify a service while its traffic enters their networks. The service identification, whether it is based on protocol identifiers or traffic pattern recognition, enables performing prioritization. As a result, some preferred applications may offer guaranteed perceptive quality by assuring bandwidth or latency. Other services use the rest of the network resources, thus being transmitted according to the best-effort model or even with administratively limited throughput.

For assessing the service (and mainly audio and video) quality, objective quality models can be categorized into two major categories, i.e., full-reference and no-reference models. While the full-reference models require a reference source to perform quality analysis on the received media and typically have a high correlation to the perceived service quality, the no-reference models do not use such references but have lower accuracy. Applying full-reference models to assessing public services is problematic. The central question is how a valid

P. Orosz, T. Skopkó and T. Marosits are with the Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics (BME), Budapest, Hungary (e-mail: orosz@tmit.bme.hu, skopko@tmit.bme.hu, marosits@tmit.bme.hu).

Manuscript received September 5, 2022;

DOI: 10.36244/ICJ.2023.1.8

reference source can be obtained. The primary disadvantages of the full-reference model are a) offline operation (i.e., post-processing), b) requirement for the original sample (as a reference) and as a fundamental privacy issue, c) analysis of user data. Based on the new methodology, we propose a real-time full-reference assessment model to identify service quality restrictions the Internet access provider applies.

Considering the assessment of network neutrality (and particularly with an application-specific focus), objective assessment methods can be categorized into three major categories based on the applied traffic pattern, i.e., real, replayed, and generated traffic. The two central questions are: i) How an assessment method relates to the real user scenarios and traffic patterns? ii) How can false detection be identified, and what is the probability of a false result? Network neutrality measurements on the public Internet cannot be considered a repeatable, all-the-way objective measurement from the metrological perspective. A comprehensive neutrality analysis covers a wide scale of measurement types that should be performed reliably on a live network. While the protocol-based test can be executed with a relatively low false ratio, application-specific measurements raise multiple methodology-related issues.

The primary goal of this paper is to establish a new assessment paradigm - real-time application- and platform-specific analysis of network neutrality on the public Internet (assessment anytime, anywhere). The involvement of public cloud services enables the elaborated objective models to continuously adapt to the ever-changing public networks and platforms by exploiting the benefits of real service traffic. Moreover, the novel measurement paradigm allows the assessment of network neutrality for a wide range of applications with a low false rate.

The remainder of the paper is organized as follows. Section II presents the related works in the field of technology- and application-specific analysis of network neutrality. Section III introduces the new methodology focusing on the novel principles, and Section IV describes the assessment method that aligns with the criteria specified in BEREC measurement directives [2] [3]. In contrast, Section V presents a proof-of-concept use case applying the method to detect video streaming restrictions. Section VI shows validation results to demonstrate the effectiveness of our model. Finally, Section VII concludes the paper.

II. Related Work

Previous works on network neutrality assessment, in general, try to detect possible differentiation with a manipulated user traffic pattern and compare the metrics of the transmissions [4] [5], being the latter the reference. Some of them aim to detect the shaper algorithm and its parameters as well [6] [7]. The real challenge for all of them is to lower the false positive ratio (differentiation detected, but there is none) and false negative (differentiation stays under the radar) detection.

NetPolice's idea uses two flows between the endpoints: the original is the reference, and a generated one with a similar timing but with different ports and payload is to be replayed [8]. The flows are transmitted among similar packet loss if no shaping is applied between the endpoints. Routing information is also considered. Since it is based on ICMP replies of the hops, its usability is affected by rate-limiting and protocol prioritization.

Glasnost uses real traffic and its randomized copies with similar timing [9]. To detect shaping, these copies are replayed via the measurement servers at the same or higher rates than the original one. Maximum application throughput is compared with application control flow throughput as well. To improve its accuracy, user traffic traces were collected and analyzed; different flow types were compared on the same network path, as well as measurements were executed at least five times and at least for 60 seconds per use case. It used a dedicated application (discontinued).

DiffProbe also uses packet replay at different rates, and link saturation [10]. It takes packet loss and end-to-end delay into account when detecting possible differentiation. Its limitation is that it also classifies by port numbers and payload type since behavior-based differentiation methods were unreliable at the time.

OONI's goal is detecting Internet surveillance and censorship [11]. Using active probing, it tries to connect various HTTP-based services. It also utilizes DNS lookup and multi-protocol traceroute, detects TCP resets, and man-in-the-middle (MITM) SSL/TLS interventions. However, more of these tests affect the application level; it does not perform application or service-specific tests.

Network tomography and inference framework use a different approach [12]. It measures TCP and UDP traffic flow congestion between a set of endpoints and constructs a linear system of equations. An unsolvable system means positive detection. However, this solution needs extensive infrastructure and a large number of users. The advantage is a low false detection rate and keeping count of TCP dynamics.

Researchers behind CONNEcT realized that identifying network neutrality measurements can help ISPs divert the results [13]. The detection is based on packet loss and passive path capacity measurement. Their solution uses a covert communication channel between measurement hosts: these metadata and samples are hidden within the application data. No application-specific measures are made, however.

WindRider focuses on measurements between its measurement servers and mobile endpoints [14]. Although measurement is lightweight, the method requires user feedback, and demands are special on OS features (packet capturing, etc.) that limit the wide implementation. The user feedback does not judge an application's QoE (Quality of Experience) itself.

Statistical analysis is commonly used to minimize the effect of background traffic as well as to reduce the false detection ratio with more or less success [15]. Only a few methods use real application traffic during the measurement, and none of them use QoE as the base of their detection. Some of the methods above require endpoints placed at other ISPs or dedicated applications or frameworks on the endpoints. If a measurement server is used, ISPs can also recognize and whitelist the measurement traffic.

III. METHODOLOGY

A. Measurement on live networks

Our measurement method approaches the detection by the real user QoE. First, the user's access parameters are determined. Then the measurement is executed with two instances simultaneously: it is measured at the user endpoint and in a container with the same access parameters at an Internet exchange point (IX). Since IX can be considered a neutral access point, the measurement results will be used as a real-time generated reference for the measurement. The measurement itself measures the specific service in subject instead of only replaying a recorded traffic pattern and measuring QoS metrics. Despite the ISP being inspected could detect the measurement control server's IP addresses, the baseline of the measurement cannot be altered.

Measurements performed on a live public network (i.e., the Internet) are not repeatable in the sense that we execute measurements in a controlled laboratory environment. Accordingly, evaluating the measurement results requires a new perspective and alternative methods. Whether the assessment involves packet-level or application-level performance metrics, consecutive measurements never give the same result, even between two designated endpoints. In order to improve the reliability of the outcome, there are existing methods to filter out traffic interference and other endpoint-related constraints [16] [17]. On the one hand, there exist measurement methods supporting a single measurement by eliminating the effect of transient events that occurred during the assessment process. Finally, there are statistical methods to handle anomalies based on a large number of measurements. When such an assessment tool is made publicly available, we must handle the most challenging scenario, i.e., the network neutrality assessment is allowed to perform by anybody using his/her computer and Internet access. The primary question this scenario raises is how we can get a reliable result, possibly free from false detection or identifying an invalid case at least. From the metrological perspective, the main task is to minimize measurement error even with these constraints. Accordingly, we will discuss the factors that affect measurement reliability and may induce false detection on a public network.

B. Handling false detection

Focusing on the root cause of false detection, we can categorize the sources of unwanted events and properties into two categories: i) traffic interference and ii) resource interference. While the former covers background traffic that is concurrently transmitted on any segment of the IP path during the measurement process, the latter refers to the host's hardware resources (e.g., CPU and system memory) that are shared between running processes, including the measurement itself. While lightweight background traffic and concurrent processes do not affect the reliability of the measurement result, heavy workloads may induce false detection of traffic manipulation. Verifying resource availability is a pivotal step to minimize the probability of false detection. We propose a preliminary measurement phase for a web-based approach that aims to estimate the available access bandwidth and the packet loss ratio. Alternatively, an application-based approach also enables to verify CPU, memory, and bandwidth availability preceding the network neutrality measurement.

C. Causes of detection uncertainties

1) Background user traffic: Background traffic may alter the time-domain behavior of the measurement traffic and thus affect the measured transmission properties. Since the presence of background traffic is inevitable in public network measurements, our goal is to minimize its effect on the measurement result. In the worst-case scenario, excessive background traffic congests a network link, resulting in packet loss in the measured traffic. These loss events directly affect the application-level performance of the measured service.

2) Process-level interference on client-side resources: Available CPU cores, system memory, and Internet access capacity are shared resources between concurrently running system and user processes.

3) Congestion on the content provider's infrastructure: Our proposal involves the content provider's infrastructure in the measurement process. Accordingly, bottlenecks occurring on the provider-side directly impact the measurement outcome. However, cloud-based virtualized content services scale well by design to maintain high service quality even in peak time periods. Meanwhile, specific scenarios may occur when service quality drops due to unexpected security or resource provisioning issues. Our proposed dual test method can identify quality degradation originating from a bottleneck that evolved in the provider's infrastructure.

D. Whitelisting the measurement traffic

Most of the discussed proposals are based on the client-server communication model. The drawback of this architecture is the static nature of the server-side, which is easily identifiable via its IP address. This enables Internet Service Providers to dynamically detect and prioritize measurement traffic at packet-level in real time.

Solution: Since the proposed method fetches real content from the content provider, the server-side measurement IP address belongs to the provider's IP network. This feature disallows ISPs to identify the measurement traffic on their networks. However, a single TCP connection permanently exists between the measurement client and the control server throughout the entire measurement session. While the server's IP address is fixed, it can be masqueraded by deploying one or multiple relay servers (exclusively for the control messaging) in an arbitrary public cloud. In this case, ISP can only identify the IP of the relay server that belongs to the public cloud provider.

IV. THE PROPOSED METHOD

A. Feature overview

Our proposal has three key novelties: i) it is horizontally scalable to any cloud-based service, ii) it measures real traffic originating from a public service provider, and iii) measurement reference is created concurrently in real time. While

we proved the feasibility of the proposal using cloud-based services, there are no technological constraints limiting its usage to web-based applications on the client side. The primary goal of the method is to perform application-specific network neutrality measurements on the public Internet reliably.

B. Measurement architecture

The architecture incorporates three major components to support the dual measurement concept: the real client, the measurement server, and the content provider service (see Fig. 1).

Every neutrality measurement is performed in parallel: the real client and its emulated twin (imitating the Internet access parameters of the real one) simultaneously measure the same service. The real client is using the neutrality measurement application on its host, and the emulated client is running the same application in the emulated container on the server.

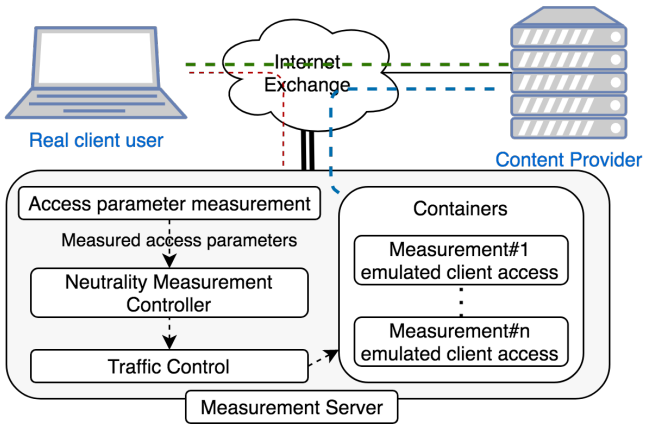


Fig. 1: The proposed measurement layout and architecture

The measurement server is a generic server architecture for running the software components required for the measurement process: performing Internet access parameter measurements for the real clients, creating network containers for Internet access emulation, executing service measurements for the emulated clients, as well as controlling the real client during the process. The server should have an uplink (marked with a double black line) broad enough for measuring all possible ISP access profile types (e.g., at least 10 Gbps if we want to measure multiple clients with 1 or 2 Gbps access). After bandwidth measurement, real clients exchange only control messages with the measurement server. This logical connection is marked with a red dashed line. The service data stream between the real client and the content provider is marked with green, and between emulated clients and the provider is marked with blue dash lines.

C. Real-time reference: dual measurement with client emulation

The most crucial component of the measurement architecture is the emulated client that adopts the principal network access properties of the real measurement client: download and upload bandwidth, round-trip time (RTT) latency, and optionally packet loss ratio. Furthermore, the measurement server should be deployed to a location independent of the user's Internet Service Provider, i.e., to the logical proximity of an Internet Exchange Point. A preliminary QoS measurement phase can determine the network access properties of the real client.

D. Measurement workflow

The Neutrality Measurement Controller (see Fig. 1) manages the measurement by instructing both the real and the emulated clients. The following phases are executed during a measurement session:

1) Preliminary QoS measurement on the real client to determine the major network access parameters.
2) The emulated client adopts the access profile of the real client by software-based emulation of the same QoS parameters within a network container.
3) Both clients initiate the measurement by requesting the same content from the content provider.
4) While the content is streamed online, both clients measure the key performance indicators. We note that the set of appropriate performance indicators should be uniquely defined for each service (see Section V for proof-of-concept use cases).
5) Both clients independently evaluate service quality based on the measured KPIs.
6) Processing the evaluation results, the measurement controller determines the overall service quality and represents it to the user on a predefined quality scale (e.g., the ITU P.800 MOS scale [18]).

The preliminary QoS measurement incorporates a method developed in one of our previous research and development projects (SCL Broadband Measurement System [1]). This system is validated and applied by the Hungarian National Media and Infocommunications Authority as a reference for measuring Internet access parameters. See Section VI for validation details.

Handling false detection: Including a real-time measurement reference, our method also enables identifying bottlenecks on the content provider-side.

V. PROOF-OF-CONCEPT USE CASES

This section introduces a video-on-demand use case for Youtube to prove that our method is feasible and applicable to a wide range of cases. Though, the neutrality measurement should be uniquely implemented for each service or platform. A measurement session starts with a quick estimation of user access parameters: upload and download bandwidth is determined, as well as packet loss and RTT. These parameters will be used to create a measurement container on a measurement host located at a neutral Internet access point. Reference measurements are then executed within this container, and it will be called emulated client. As mentioned previously, we

[1] https://szelessav.net/en/

examine the service's quality at the real and the emulated client in parallel.

We had to look for the measurable and gradable phenomenon of the use case to correlate them with QoE. However, we should only consider phenomena whose cause is the QoS degradation of the user's Internet access or degradation of the inspected service itself.

Examples of such a phenomenon: playout buffer becomes empty, video/audio stream quality changes (degrades), video/audio stream source bitrate changes (decreasing bitrate during adaptation), and video/audio decoding errors during the playback.

As we measure services in the browser at the user endpoint, we had to rely on the streaming service's API (Youtube, in this case). We chose the iFrame Player API [19] because it provides functions for measuring the phenomena mentioned above.

The following functions and events can be considered helpers for the measurements:

- player.getVideoLoadedFraction():Float – queries the downloaded ratio of the entire stream.
- player.getCurrentTime():Number – queries the seconds elapsed since the start of the playback.
- onPlaybackQualityChange – the event triggered when (the class of) playback quality changes.

Accordingly, we can construct a metric based on the ratio of the time when the playback buffer is empty against the entire playback time. Alternatively, the metric can express the time ratio of the degraded playback quality against an explicitly requested media quality (e.g., playing back in SD instead of the requested 720p). Then, we can apply a weighted linear combination of them.

We also had to consider the media playback process in the user's browser. Each playback starts with a pre-fetch phase when the application loads the first part of the video stream to the playout buffer, then it switches to a playback phase. Supposing that the user equipment does not have resource constraints, the playback only gets stalled when the playout buffer becomes empty. We can query the downloaded ratio of the stream by the *player.getVideoLoadedFraction()* function at any time. This function reports the downloaded proportion independent of the video quality and size. Since we know the video's duration, we can transform the loaded fraction value into a time value to express the time position until the video was already downloaded. We will call this *video time*. Of course, parts of this downloaded content are already decoded and rendered, but another part of the undecoded bitstream may reside in the playout buffer. The *player.getCurrentTime()* function reports the time elapsed since the playback started. If we look at the time function of these two metrics, a continuous playback's time can be depicted by a line, but the loaded amount (video time) is a monotonic growing curve, as seen in Fig. 2.

We marked the buffer status in green and the playback time in red. The buffer becomes empty when playback time is not strictly monotonically increasing or rising above the video time. In the presented scenario the green curve is always above the red; that is buffer never gets empty.

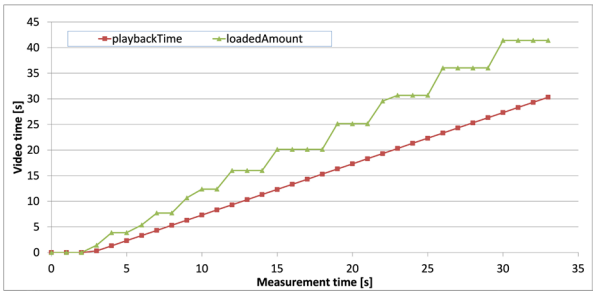


Fig. 2: Relation between playback time and video time on a 4/1 Mbps connection

By continuously sampling the playback process, the Youtube playback QoE can be estimated by the number of samples with continuous playback (playout buffer never gets empty) against the total number of samples, as of (1). Here, the playback QoE indicates network neutrality, i.e., Network Neutrality Index (NNI).

$$NNI^{1}_{str} = \frac{\text{\# of samples of continuous playback}}{\text{total \# of samples}} \quad (1)$$

Besides the measurements executed locally in the user's web browser, we can indirectly rely on the Youtube service's download controller reports. These reports aim to inform the content provider about the users' communication environment and help optimize the download process to improve the QoE. One feature of this optimization is the automatic selection of the bitstream format. The download controller is notified via the *onPlaybackQualityChange* API event. The event reports a quality class string, but indirectly refers to the video bitstream format. The class identifies a subset of bitstream formats, e.g., hd720p includes bitstream formats encoded with various profiles of avc1 and vp9, all being 720p resolution. Thus, we cannot determine the current bitstream format, bitrate, framerate, etc., or the exact source data rate. Still, we have received an explicit notification about the client's Youtube download controller; what is the best achievable quality class that the client can receive besides the current (previously measured) access parameters. A change in quality can denote a change in available bandwidth and thus can be considered a service quality metric. This event hook also completes our estimation method without being limited to periodic sampling.

Fig. 3 shows the relationship between the abovementioned status information. The graph also shows that decoding the video content (i.e., rendering) can cause a performance bottleneck at the user endpoint, limiting the achievable quality, even though the quality of raw data transmission is appropriate. This test was executed at the bandwidth of 100 Mbps in both directions, while the 8K video content's source data rate does not exceed 25 Mbps. Thus, available access link bandwidth was not a limiting factor. We also repeatedly experienced stalled downloads and noticed that the download controller switched to a lower-quality class. For example, on Fig. 3 at 33 seconds, the playback stopped (see red graph), then the buffer was purged, and after that, from 38 seconds, the buffer

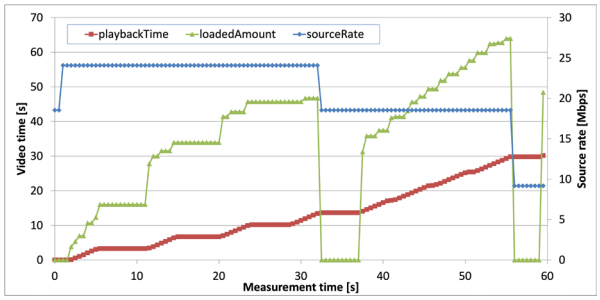


Fig. 3: Symptoms of video quality class change in function of time on a 100/100 Mbps connection

size started to increase again (see green graph) but the source rate - which relates to the video quality - was lower than before (see blue graph).

The *onPlaybackQualityChange* event occurrs whenever the quality class is changed during the playback, allowing us to perceive the change immediately. This concludes with a more precise reconstruction of the changes during the playback. We can derive a metric from these series of events providing the basis for the network neutrality index of the Youtube video streaming service. A simple case of the metric relating to the violation of network neutrality is presented by (2). The target format is the same or better class than the auto-selected one during the playback start. Accordingly, the metric is calculated by the number of good-quality samples (i.e., that are not scaled down) against the total number of samples.

$$NNI^2_{str} = \frac{\text{\# of samples in target format}}{\text{total \# of samples}} \tag{2}$$

We can construct measurement results from any of the metrics discussed previously. For a more precise estimation, we decided to use the sum of equally-weighted normalized values as of (3).

$$NNI_{str} = 0.5 \times NNI^1_{str} + 0.5 \times NNI^2_{str} \tag{3}$$

Of course, the same formula must be used for both the real and the emulated client. Suppose the reference (emulated client) value is lower than the real client's one. In that case, we can suspect that degradation is not caused by the user's Internet access but a bottleneck at the content provider. To create an index value that is easier for humans to interpret, the NNI value can be formed between 1 and 10 as of (4).

$$NNI^{human}_{str} = 10 \times (0.1 + 0.9 \times NNI_{str}) \tag{4}$$

VI. Validation

The proposed method relies on measuring the network access properties of the client and applying it to the emulated client to concurrently access content available at the same content/service provider. Accordingly, the performance of the QoS emulation directly impacts the validity of the measurement reference provided by the emulated client. Since the network emulation incorporated download and upload bandwidth and latency, the primary aim of the validation is to compare the traffic properties of the real and emulated clients. This section presents the validation results of the QoS emulation. We can measure the effectiveness of the emulation by analyzing the similarity between the two traffic patterns. In our proof-of-concept system, the emulation subsystem is based on the Linux kernel-based Network Emulator (NetEm).

Three criteria of the validity: 1. Independent Internet access (Internet Exchange), 2. Valid QoS emulation of the access properties, 3. Valid measurement reference.

Our method uses a trusted reference measurement executed parallel with the user's measurement during network neutrality-related service inspection. The measurement hosts use container technology to emulate the user's access parameters properly. Our method can be considered valid if the emulation is working correctly. Access parameters are measured on a link without background traffic. Bandwidth, bi-directional latency (RTT), jitter, and packet loss are measured.

Our proposal introduces a novel application-specific measurement method incorporating a unique approach for the measurement architecture as well. In contrast, previous works on network neutrality assessment, in general, try to detect possible differentiation with a manipulated or generated user traffic pattern. Therefore, they cannot be effectively compared to our proposed solution.

A. Measurement setup

For validation, a setup with two hosts connected directly was used. One host is a bandwidth measurement system, and the other one runs the browser in a container with specific access parameters. Both hosts are capable of generating traffic of at least 1 Gbps.

Fig. 1 details the measurement system's structure. The dedicated container is created each time a measurement has to be run. At the beginning of the user's measurement, a quick quality of service assessment is run on the client side. This phase determines its network access parameters (download and upload bandwidth, delay, and packet loss rate). These access parameters are used to create a similar environment for the neutrality measurement at the initialization of the container.

The container is based on Linux Network Namespaces (netns). It is a Linux kernel technology builder for software containerization. A netns container has its own network interface but can be run normally with the same computing resources anyway. A point-to-point network connection is established between the system and the container.

Linux Traffic Control (tc) is a generic tool used to configure shaping and policing but also supports scheduling and dropping. The proper configuration using the *tc* command can emulate a link with specific access parameters. Since the point-to-point connection is built between two virtual network devices, each direction can be configured independently; thus it can also emulate an asymmetric connection.

For the more sophisticated realization of specific network access, we applied Linux NetEm (netem) scheduler since it allows us to add delay, jitter, and packet loss.

After the container is set up, the neutrality measurement is run simultaneously on the real client and the reference

container. If the network access is emulated correctly and the client has neutral Internet access, similar results should be obtained from both parties.

B. Validation methodology

During the measurement sessions, we iterate through a set of access parameters. After a session, the measured results are compared to the nominal access parameters: average and deviation are calculated to parameters bandwidth, delay, and packet loss. We expect the values to be within an error margin of 4%.

RTT is measured by generating consecutive websocket ping-pong messages. These packets are sent within TCP packets with an urgent flag and are replied to immediately by the receiver's websocket stack. Minimum and maximum RTTs are determined, as well as the average and the standard deviation of delays.

Throughput is measured by saturating the link in the affected direction by using the appropriate number of parallel websockets. The transfer is started with only one socket to avoid overloading narrow links. If per-socket throughput is over a target rate, new sockets are opened to try to saturate the link.

Packet loss is measured using TCP analysis: TCP segments are traced, and if a segment or its part is re-transmitted, it is calculated as lost. We derive packet loss from the amount of re-transmitted data against the total bytes transferred.

C. Container validation

The quality of service parameters of the connection seen by the container was validated with the Broadband Measurement System of the National Media and Infocommunications Authority[2]. Thus, we proved that the connection available for a container created in a net-neutral environment is equivalent in terms of quality of service to the connection that the user of the net-neutral metering system receives from its service provider. The accuracy of this Broadband Measurement System and the stability of the measurement series under laboratory conditions were validated by measurements at known native interface speeds and by using the Spirent Attero X network emulator as a bandwidth and delay reference. According to the measurements, the error of the internet speed measurements is under 2% between 100 kbps and 2 Gbps if the round-trip connection delay does not exceed 100 ms or the BDP (Bandwidth Delay Product) does not exceed 4 MB.

1) The correct value for the object to be measured: In the tests, we use maximum-sized Ethernet frames without 802.1Q VLAN tags to measure throughput, so their length is 1538 bytes in the physical layer. Because each packet contains a 12-byte TCP option after the TCP header, the maximum length of the TCP segment embedded in the IP packet within the Ethernet frame is only 1448 bytes. Overall, the throughput can be at most 1448/1538 = 94.1482% of the physical bit rate at some native interface speed if the IFG (Inter-Frame Gap) is not reduced. Suppose the speed limit is set with NetEm, and this limit is interpreted in the data link layer. In that case, the theoretical maximum is 1448/1514 = 95.6407% of the set value if we do not consider the possible reduction of the IFG. The presented measurements were performed uniformly over HTTPS.

Regarding the delay time, it should be noted that when creating a packet delay on the prepared virtual interface, NetEm even adds to the set value the time it takes for a packet of the same size to pass through an interface with the same physical speed. On HTTPS, we use 119-byte Ethernet frames from either the client or server for delay measurement. This would result in another 24 bytes in the physical layer (IFG + Preamble + Start-of-Frame-Delimiter + CRC), but this should not be considered, as NetEm's rate limiter settings are interpreted in the data link layer. Thus, the combined packet forwarding delay of packets passing through the interface in one direction or another, for example, on a 1 Mbps symmetrical connection 1.904 ms, which corresponds quite well to the excess experienced, so we can correct the value of the measured delay with it. In the higher speed range, the effect is no longer significant.

2) Measurement results: In addition to the mean (m) and standard deviation (σ) of the bidirectional delay, our data include the mean and median of the cleaned set of samples per second of download and upload speeds in the transport layer, as well as the packet loss rate detected in each direction. For us here, the mean and standard deviation of the bidirectional delay and the downstream and upstream throughput will be the most interesting. The following is a table-form presentation of some of the measurement data, namely the mean and standard deviation of the measurement results. At least 50 measurements were made at each setting.

The presented results in Tables I-III prove that we can provide a network connection to the reference container in a predictable way according to the interface emulation settings. Table I provides the statistics of the measurements of a container with 1 Mbps symmetric bandwidth, Table II presents similar statistics of a container with 100 Mbps symmetric bandwidth. In contrast, Table III provides the statistics of 1 Gbps symmetric bandwidth container. These bandwidths were set in the NetEm. The first column of the tables shows the intended bidirectional delay, also set in the emulator.

Table headers shorten the followings:

- D^{set} is the delay in *ms*, which was set as the interface delay of the container's virtual interface. This delay was symmetrically distributed.
- RTT^{meas}_{avg} is the average of measured end-to-end delay samples on the container's virtual interface in *ms*.
- RTT^{meas}_{stdev} is the standard deviation of measured end-to-end delay samples on the container's virtual interface in *ms*. This can be considered as the delay jitter.
- $DSrate_{avg}$ is the average downstream rate on the container's virtual interface in *Mbps*.
- $USrate_{avg}$ is the average upstream rate on the container's virtual interface in *Mbps*.

While $DSrate_{avg}$ and $USrate_{avg}$ are measured in the transport layer, they represent the TCP throughput the user in each direction can achieve.

[2]https://szelessav.net/en/

The cells of Tables I-III contain two numbers. As it is mentioned in the 2nd column of the tables, the upper number is the average of the measured quantity identified by the header of the actual column. In contrast, the lower number is the standard deviation of the same quantity. The averages in columns 3, 5 and 6 can be compared to the values preset in the emulator considering the argumentation mentioned above about the correct value for the measured object.

TABLE I
STATISTICS OF THE MEASUREMENT RESULTS AT 1 MBPS

D^{set}		RTT^{meas}_{avg}	RTT^{meas}_{stdev}	$DSrate_{avg}$	$USrate_{avg}$
0	m	2.277667	0.089067	0.961765	0.952292
	σ	0.029331	0.053768	0.00616	0.00168
1	m	3.271317	0.097783	0.960243	0.952468
	σ	0.038381	0.042514	0.007081	0.001556
4	m	6.279983	0.085167	0.958726	0.952579
	σ	0.029907	0.187768	0.006517	0.001949
10	m	12.2851	0.08595	0.958213	0.95258
	σ	0.024229	0.057395	0.007618	0.001532
30	m	32.28538	0.1236	0.958262	0.952457
	σ	0.032863	0.217209	0.007157	0.001705
100	m	102.3016	0.089153	0.958991	0.953304
	σ	0.015433	0.055426	0.00711	0.00142

TABLE II
STATISTICS OF THE MEASUREMENT RESULTS AT 100 MBPS

D^{set}		RTT^{meas}_{avg}	RTT^{meas}_{stdev}	$DSrate_{avg}$	$USrate_{avg}$
0	m	0.205198	0.034887	95.65311	95.56265
	σ	0.011178	0.004756	0.037675	0.012557
2	m	2.388895	0.039735	95.65777	95.55468
	σ	0.026957	0.061805	0.048672	0.014351
4	m	4.394665	0.054578	95.64517	95.55992
	σ	0.033135	0.106249	0.019685	0.014976
16	m	16.39481	0.042604	95.64445	95.54124
	σ	0.022767	0.032298	0.020738	0.01707
30	m	30.39306	0.046067	95.63466	95.52246
	σ	0.019898	0.026699	0.018684	0.027681

TABLE III
STATISTICS OF THE MEASUREMENT RESULTS AT 1 GBPS

D^{set}		RTT^{meas}_{avg}	RTT^{meas}_{stdev}	$DSrate_{avg}$	$USrate_{avg}$
0	m	0.087567	0.023533	956.8405	956.4033
	σ	0.012218	0.007524	0.784973	0.225397
1	m	1.151567	0.025517	956.7349	956.31
	σ	0.026311	0.009899	0.617618	0.19924
2	m	2.19315	0.0341	956.7329	956.3524
	σ	0.028002	0.03899	0.646558	0.18775
4	m	4.197133	0.042383	956.7777	956.4288
	σ	0.022469	0.055571	0.810262	0.284104
10	m	10.19637	0.03395	956.8733	956.8491
	σ	0.021568	0.013686	0.65082	1.374928
16	m	16.19535	0.0404	956.7959	956.3924
	σ	0.014769	0.046178	0.713448	0.26032
20	m	20.1927	0.037117	956.7736	956.2828
	σ	0.017342	0.014367	0.848646	0.169963

We can see that the rates are very close to their theoretical limits, which shows, that the emulation is very accurate. This statement is confirmed by the values of sigmas (σ) which are under 1% of the appropriate averages (m) at 1 Mbps and under 0.1% at 100 Mbps and 1 Gbps. The difference between the measured and the expected RTT - note that this later differs from the delay set in the emulator - is no more than 2%. Obviously, except for the delay set to 0, where the standard deviation of the average RTT can be higher. Column 4 presents the average and the standard deviation of the delay jitter series in the tested scenarios. Although NetEm would allow it, we do not use this as a configuration parameter of the container, but we measure it. As we can see, the average delay jitter is less than 1 ms in every scenario, which is considerable for real-time multimedia services and these values are stable.

As a summary of the container validation, it can be established that for both the statistics presented here and those omitted due to lack of space support that, we can infer the measurement result from the given settings with high reliability, i.e., the network emulation is accurate. The standard deviations show that the results are very stable.

D. Validation of the VoD (Youtube) assessment method

Two different measuring devices - in this case, the arbitrary browser used as a client and the Chromium browser engine running in the node.js environment in the container - can be considered the same if the distribution of their measurement results is identical for a measured quantity. Instead of matching the distribution, we can also accept the sameness of the statistical indicators, provided that sufficient measurements are made.

Among the features presented earlier, the download rate returned by the *player.getVideoLoadedFraction()* function is what gives a monotonically increasing curve over time and can be considered a linear function of time. This offers the option of fitting a line to this with the least-squares error, as Fig. 4 shows. In the case of measurement, the fitted line can be given by its intercept and slope, while the measuring device is characterized by the statistical properties of the slope and intercept of the regression lines of the measurements in the measurement series. Based on the average of the axis intersections and the average of the slopes, we can draw a line - this will be referred to later as the "center line" - which can actually be interpreted as a measure of the current client configuration. However, we can derive the uncertainty of this measurement based on the standard deviations of the axial intersections and slopes. If we add or subtract twice the standard deviation of the mean slope and add or subtract twice the standard deviation from the average slope, we get two more lines, later referred to as "bounding lines", which delimit it on both sides. The part of the plane to which the fitted line of the individual measurements falls with a probability of 95%. In fact, we gave the characteristic of the current configuration of the measuring instrument, as shown in Fig. 5, where the center line is drawn in blue, and the upper and lower boundary lines are drawn in red, and green, respectively.

1) Dependence of the reference client's parameters on the operating system: Since Youtube traffic is over TCP[3]. There is a measurable difference between TCP implementations on different operating systems (e.g., Ubuntu Linux and Windows 10). So it is expected that the measurement results of the video streaming service testing module of the net-neutrality measurement system will also be affected by the client's

[3]Youtube also supports QUIC/UDP, but the container limitations related to Layer 2 traffic, thus they can be considered with the same impact in the case of UDP. The API used for the Youtube-measuring method mentioned above is independent of the transport protocol.

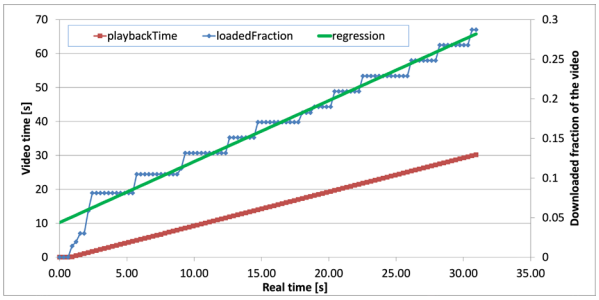


Fig. 4: Download rate curve with the line fitted to it (50/20 Mbps, Firefox/Ubuntu)

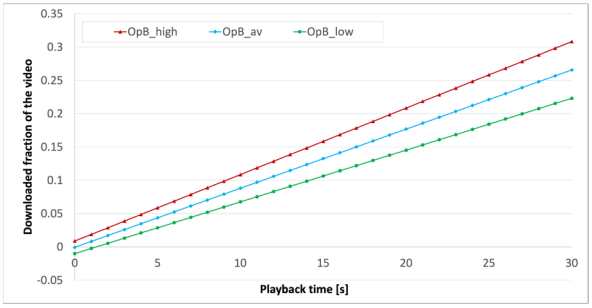
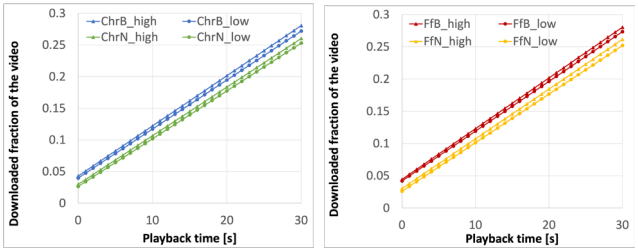


Fig. 5: Characteristics of the 100/100 Mbps connection (Opera/Ubuntu)

operating system. This permanent discrepancy is unavoidable because node.js runs the Chromium browser engine in the container replicating the client. However, based on our studies, this deviation is linear. Based on the information available in the measurement system, the permanent error can be reliably estimated, and then the measurement result can be corrected. In our experience, the extent of this discrepancy also depends on which browser, possibly which version, is used and the bandwidth of the connection available to the user.

2) Dependence of the reference client's parameters on the browser: In Fig. 6 below, we can see that when testing on Ubuntu Linux over a 100 Mbps/100 Mbps connection, both the Chrome browser and the Firefox browser outperform the container. However, it is also noticeable that the behavior of browsers is very similar to each other. Also, on Ubuntu, for example, looking at a 30 Mbps/10 Mbps connection, we see that the characteristics of the browser and the container get closer together whether we use Chrome or Firefox, so the difference is speed-dependent.

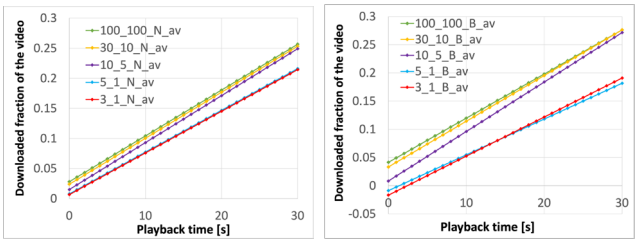
3) Dependence of the reference client's parameters on the Internet access speed: Since the most advanced video streaming providers are constantly sampling the maximum available speed of the available network connection, the behavior of the download controller they use changes as the available capacity begins to approach the source speed of the video stream from above. As mentioned above, the Chrome browser outperforms the container at higher speeds on Ubuntu Linux. However, as the speed decreases, this advantage decreases, then disappears and eventually becomes a disadvantage. A similar phenomenon can be experienced using Firefox. However, there



(a) Chrome (b) Firefox

Fig. 6: Browser and container characteristics (100/100 Mbps, Chrome/Ubuntu and Firefox/Ubuntu)

are much more minor differences between the cases, so for ease of interpretation, Fig. 7 shows the characteristics of the Chrome browser and the container at various access speeds (100/100, 30/10, 10/5, 5/1 and 3/1 Mbps, respectively). For transparency, the characteristics are now represented not by the delimiting lines but by their center line.



(a) Container (b) Browser

Fig. 7: Chrome browser and container characteristics on different speeds on Ubuntu

VII. CONCLUSION

From an Internet provider's perspective, managing network resources is a crucial operational task, guaranteeing the quality of experience for cloud services that dominate their traffic mix. However, its endeavor to control resources, i.e., traffic engineering based on service popularity, may offend the original best-effort paradigm of the global Internet. This paper presented a real-time full-reference objective method to assess network neutrality with application awareness. The proposed method may support the Regulation (EU) 2015/2120 with scientific fundamentals. Furthermore, the expected scientific results can indirectly support EU citizens to objectively assess network quality by opening the way towards a new generation of network neutrality measurement tools for national communications authorities. The method incorporates three novelties to support the user-centric analysis of potential restraints affecting public services on the Internet: i) it supports application-specific measurements and involves real content and traffic, ii) the measured traffic originates from the content provider's cloud infrastructure, iii) the reference is created in real time. Accordingly, the paper also proposed a novel measurement layout. We have validated the incorporated client emulator and demonstrated the feasibility of the measurement

method with a video-on-demand use-case using an extensive set of laboratory measurements. In the future, we would like to extend the measurement capability of the implementation to a broader range of cloud services. Furthermore, the Hungarian National Media and Infocommunications Authority is evaluating the current version of the measurement system. It will potentially be available for the public to perform web-based network neutrality measurements.

REFERENCES

[1] EU, "Regulation (eu) 2015/2120 of the european parliament and of the council," *Official Journal of EU L310*, 2015.

[2] BEREC, "Berec net neutrality regulatory assessment methodology," *BEREC BoR (17) 178*, 2017.

[3] BEREC, "Berec net neutrality measurement tool specification," *BEREC BoR (17) 179*, 2017.

[4] M. B. Tariq, M. Motiwala, and N. Feamster, "Nano: Network access neutrality observatory," Georgia Institute of Technology, Tech. Rep., 2008.

[5] G. Lu, Y. Chen, S. Birrer, F. E. Bustamante, and X. Li, "POPI: a user-level tool for inferring router packet forwarding priority," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 1–14, 2009. DOI: 10.1109/TNET.2009.2020799

[6] P. Kanuparthy and C. Dovrolis, "ShaperProbe: end-to-end detection of ISP traffic shaping using active methods," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 473–482. DOI: 10.1145/2068816.2068860

[7] U. Weinsberg, A. Soule, and L. Massoulie, "Inferring traffic shaping and policy parameters using end host measurements," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 151–155. DOI: 10.1109/INFCOM.2011.5934941

[8] Y. Zhang, Z. M. Mao, and M. Zhang, "Detecting traffic differentiation in backbone ISPs with NetPolice," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, ser. IMC '09. New York, NY, USA: Association for Computing Machinery, Nov. 2009. ISBN 978-1-60558-771-4 pp. 103–115. DOI: 10.1145/1644893.1644905

[9] M. Dischinger, M. Marcon, S. Guha, P. K. Gummadi, R. Mahajan, and S. Saroiu, "Glasnost: Enabling End Users to Detect Traffic Differentiation." in *NSDI*, 2010, pp. 405–418.

[10] P. Kanuparthy and C. Dovrolis, "DiffProbe: Detecting ISP service discrimination," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9. DOI: 10.1109/INFCOM.2010.5461983

[11] A. Filastò and J. Appelbaum, "OONI: Open Observatory of Network Interference," in *FOCI*, 2012.

[12] Z. Zhang, O. Mara, and K. Argyraki, "Network neutrality inference," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 63–74, 2014. DOI: 10.1145/2740070.2626308

[13] A. Maltinsky, R. Giladi, and Y. Shavitt, "On Network Neutrality Measurements," *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 4, pp. 56:1–56:22, May 2017. DOI: 10.1145/3040966

[14] "WindRider - A Mobile Network Neutrality Monitoring System." [Online]. Available: https://users.cs.northwestern.edu/~ict992/mobile.htm (Accessed: 2021-02-28).

[15] X. Castoreo, P. Maillé, and B. Tuffin, "Weaknesses and Challenges of Network Neutrality Measurement Tools," in *2020 16th International Conference on Network and Service Management (CNSM)*, Nov. 2020, pp. 1–5. DOI: 10.23919/CNSM50824.2020.9269077

[16] E. W. Chan, X. Luo, and R. K. Chang, "A minimum-delay-difference method for mitigating cross-traffic impact on capacity measurement," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, ser. CoNEXT '09. New York, NY, USA: Association for Computing Machinery, Dec. 2009. ISBN 978-1-60558- 636-6 pp. 205–216. DOI: 10.1145/1658939.1658963

[17] M. Li, Y.-L. Wu, and C.-R. Chang, "Available bandwidth estimation for the network paths with multiple tight links and bursty traffic," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 353–367, Jan. 2013. DOI: 10.1016/j.jnca.2012.05.007

[18] ITU, "Mean opinion score (MOS) terminology," Recommendation ITU-T P. 800.1. *ITU-T Telecommunication Standardization Sector of ITU Geneva*, 2016.

[19] "YouTube Player API Reference for iframe Embeds," publication Title: Google Developers. [Online]. Available: https://developers.google.com/youtube/iframe_api_reference (Accessed: 2021-09-23).

Péter Orosz is an associate professor and the head of Smart Communications Laboratory at the Department of Telecommunications and Media Informatics, BME Hungary. He received his Computer Science master degree in the field of software engineering (2003) and Ph.D. in infocommunication systems (2010) at the University of Debrecen, Hungary. His research interest covers communication networks, network and service management, QoS-QoE managed networks, online QoE prediction for media services, and acceleration of network functions.

Tamás Skopkó is an assistant professor and a member of Smart Communications Laboratory at the Department of Telecommunications and Media Informatics, BME Hungary. He received his M.Sc in in field of software engineering (2002) and Ph.D in infocommunication systems (2017) at the University of Debrecen, Hungary. His research interest covers communication networks, measurement of service quality, online QoE prediction for media services, virtualization and softwarezation of network functions.

Tamás Marosits is an assistant professor and a member of Smart Communications Laboratory at the Department of Telecommunications and Media Informatics, BME Hungary. He received his M.Sc. (1996) and Ph.D. (2012) in the field of electrical engineering at the Budapest University of Technology and Economics, Hungary. His research interest covers communication networks, service quality measurement, online QoE prediction for media services, and statistical analysis of extremely large test result sets.