Using 5G QoS Mechanisms to Achieve QoE-Aware Resource Allocation

Marcin Bosk[†*], Marija Gajić[‡*], Susanna Schwarzmann[§], Stanislav Lange[‡], Riccardo Trivisonno[¶], Clarissa Marquezan[¶], Thomas Zinner[‡]
[†]Technical University of Munich, Department of Informatics, Chair of Connected Mobility
[‡]NTNU - Norwegian University of Science and Technology, Department of Information Security and Communication Technology
[§]Technische Universität Berlin, Chair of Internet Network Architectures
[¶]Huawei Technologies Munich

***Abstract*—Network operators generally aim at providing a good level of satisfaction to their customers. Diverse application demands require the usage of beyond best-effort resource allocation mechanisms, particularly in resource-constrained environments. Such mechanisms introduce additional complexity in the control plane and need to be configured appropriately. Within 5G mobile networks, two new mechanisms for QoS-aware resource allocation are introduced. While QoS Flows enable specifying various QoS profiles on a per flow granularity, slices are dedicated virtual networks, strongly isolated against each other, with aggregated QoS guarantees. It is, however, unclear how QoS Flows and network slicing can optimally be exploited to ensure a high customer QoE while efficiently utilizing the available network resources. We address this research question and evaluate the outlined interplay using the OMNeT++ simulation environment in a multi-application scenario. We show that resource isolation induced by slicing may negatively affect application quality or system utilization, and that this impact can be overcome by fine-tuning the system parameters.**

***Index Terms*—QoE, QoE-aware networking, 5G, network slicing, mobile networks**

I. Introduction

The applications running on top of today's networks are diverse in terms of their network requirements. While, for instance, Video on Demand (VoD) streaming services have high bandwidth requirements, Voice over IP (VoIP) applications are very sensitive to delay. Driven by business incentives, network operators try to provide a satisfactory application experience to their customers by efficiently fulfilling the heterogeneous service demands. Thereby, they often rely on specific mechanisms to assure Quality of Service (QoS) guarantees in order to satisfy the user perceived Quality of Experience (QoE) [1], [2]. However, such mechanisms come at the cost of more complicated control planes, i.e., they are more costly, more error-prone, and have the potential to reduce the scalability of the control plane. Hence, they are either realized in controlled environments, on aggregate level, or within the access network. The effect of increased complexity is of particular interest for per-flow QoS mechanisms in mobile networks like EPS bearers in 4G/LTE and QoS Flows in 5G. Resource allocation is then enforced by the radio scheduler which tries to optimally share physical resource blocks among all user devices with respect to the devices' Signal-to-Noise ratio and the per-flow QoS guarantees. By making use of network slicing as introduced with 5G, the complexity of the radio scheduler can be reduced by aggregating flows with the same or similar QoS demands. More specifically, network slicing allows to run several virtual networks on the same physical infrastructure and to isolate resources between the slices. When providing one dedicated slice for similar flows with similar QoS demands, this homogeneity of flows may be exploited to reduce the complexity of the scheduler.

While the concept of network slicing sounds promising in terms of reducing the system complexity, its implications on the system's efficiency and its applicability for QoE-driven resource allocation, in combination with per-flow QoS guarantees, are not yet explored. This paper is a first step towards analyzing the potentials of these concepts, as it can be realized with a 5G network to support QoE-aware networking in a multi-application environment. For that, we consider a set of five heterogeneous applications and propose a per-application slicing approach, which we compare against several benchmarks, ranging from simple best-effort approaches to more advanced per-flow QoS settings. Our evaluations show that the system load can be increased without sacrificing the QoE, if the slices and per-flow guarantees are properly dimensioned.

On a conceptual level, our contributions include the comparison of different paradigms, ranging from simple best-effort traffic treatment to per-application QoS Flows with and without QoE-awareness. Furthermore, we provide quantitative results that allow us to derive appropriate settings for QoS Flows and illustrate the impact of the proposed mechanisms on system utilization and QoE. Finally, we make our simulation environment publicly available[1] to the research community.

The rest of the paper is structured as follows. Section II provides background information and presents related work. Section III describes our proposed approach and its control knobs. Afterwards, we describe our methodology in Section IV and the conducted evaluations in Section V. Finally, the paper concludes with Section VI.

[*]Marcin Bosk and Marija Gajić contributed equally to this paper.

[1]https://github.com/fg-inet/5g-qoe-awareness

978-3-903176-36-2 ©2021 IFIP

II. Related Work and Background

This section provides information related to the technical background of QoS management in mobile networks and outlines publications related to our studies.

A. Network Slicing

5G network technology represents an on-going research area with many stakeholders (e.g. network operators, Over The Top (OTT) service providers, and users). In [3] the authors presented a tutorial overview of 5G systems with respect to standardization and deployment challenges from a user-centric perspective. Together with SDN and NFV, network slicing was identified as a key component allowing for flexibility and a wide range of different types of service level agreements. In [4] the authors discuss fundamental challenges of an end-to-end approach for slice isolation in 5G. The challenges include the need for standardized methods for the design of isolated slices and a MANO system capable of supporting heterogeneous multi-service and multi-vendor networks. A survey presented in [5] provides an overview of the existing principles and enabling technologies as well as solutions for each part of the 5G system. In [6], the authors explore the usage of SDN/NFV for network slicing in 5G networks. They propose a high level SDN/NFV architecture serving as a slicing enabler. The authors identify fulfilment of performance isolation in a common infrastructure as a complex future-work task.

B. QoS- and QoE-Awareness

Within the last couple of years, network slicing has widely been discussed in the context of QoS-awareness. A QoS-aware network slicing framework for services with diverse QoS-requirements is proposed in [7]. The authors design an API to facilitate RAN and Core slicing to meet different services' demands, e.g low latency for tele-medicine. The authors demonstrate the applicability and prospects of using network slicing to enable the QoS guarantees for mission critical services. However, the authors only consider the per-hop delay as a QoS metric, and acknowledge the QoE-aware approach as a next research direction.

The flexible allocation of network resources in radio and core networks for the user- and control-plane so as to meet their QoS needs is one of the targeted key functionalities defined in the 5G NORMA project.[2] While the upcoming requirements and challenges are presented in [8], the work in [9] focuses on the technical solutions needed to address them and to establish the network slicing paradigm as targeted by 5G NORMA. The key elements of the architecture are Software Defined Mobile Network Controller (SDM-C) and Software Defined Mobile Network Coordinator (SDM-X). These controllers are responsible for managing the network functions within a slice as well as the multiplexing gain over shared components. Even though the authors present an architecture with NFV orchestration that ensures an efficient utilization of resources and provides support for slice mobility while keeping a satisfactory QoS/QoE level, there is no implementation or performance evaluation of such an architecture.

A first approach towards quantitatively assessing the impact of network slicing on QoE is presented in [10]. The authors propose a service assurance framework which is based on the ETSI MANO architecture. It allows to map the infrastructure performance to QoE, so that QoS- and QoE-aware management operations can be performed. With testbed-based measurements, the authors evaluate the applicability of their framework for web browsing and video streaming.

C. QoS Management in Mobile 5G Networks

In the context of 5G, QoS handling can be performed by means of QoS Flows. According to [11], QoS Flows constitute the finest level of granularity regarding differentiated treatment of traffic. Each QoS Flow is mapped to a QoS profile that defines appropriate attribute values for the corresponding application that is carried by the flow. In particular, two main types of flows are defined: Guaranteed Bit Rate (GBR) and non-GBR flows [12]. While the latter does not provide any rate-related guarantees, the former comes with two parameters to control the guarantees. These parameters include the Guaranteed Flow Bit Rate (GFBR) and the Maximum Flow Bit Rate (MFBR) which define a guaranteed minimum and a hard upper bound on the bitrate, respectively. Since throughout this paper we will have guaranteed and maximum bitrates for both flows and slices, we use GBR and MBR abbreviations instead of the GFBR and MFBR.

In order to satisfy the requirements of increasingly heterogeneous applications, two approaches can be followed. The first option consists of exploiting the QoS features that are defined in the aforementioned standards and therefore implementing complex resource reservation and scheduling mechanisms that support the outlined QoS profiles. Alternatively, resource over-provisioning can simplify traffic handling while satisfying QoS requirements. The two approaches represent trade-offs in terms of their cost and resource efficiency as well as the complexity of traffic handling.

III. Technical Realization & Control Knobs of Proposed Approach

QoS Flows allow to set a GBR, ensuring that a flow obtains at least that specified bitrate, and they allow to allocate an MBR value which constitutes an upper limit up to which the flow can borrow from other flows that are currently not fully exploiting their GBR. Network slicing enables a virtual separation of networks on the same physical infrastructure and allows to isolate resources between slices. While it is possible to use QoS Flows *within* slices, i.e., a flow can borrow from other flows within the same slice, we assume that the isolated resources between the slices do not allow for borrowing *between* slices. Hence we confine in this work on hard isolation, while according to 3GPP specifications, different degrees of isolation are in general feasible [13].

While this work evaluates the performance of both concepts on its own, we also study the implications of combining

[2]https://5gnorma.5g-ppp.eu/. Accessed 31.07.2021.

them so as to reap the benefits of both approaches. In the corresponding scenarios, we group all flows of the same QoS profile - in our case the same type of application - into one slice. Although we are still capable of performing fine-grained per-flow control, the increased inner-slice homogeneity reduces the complexity that needs to be handled by the scheduler and improves the network manageability. In addition to these management-related benefits, this increased homogeneity can potentially also be exploited for improving the network performance, e.g., in terms of the system load or the delivered QoE. All flows within a slice - belonging to the same application type - share the same QoS requirements and react in a similar way to QoS fluctuations. We exploit this fact by determining adjusted per-slice and per-flow settings, specifically tailored to the applications' characteristics. Instead of relying on pre-defined per-flow QoS guarantees, we leverage the slice homogeneity and determine bitrate guarantees and borrowing capabilities so as to allow each application to reach a given QoE target.

For our evaluations, i.e., tuning the slice- and flow-specific settings in a QoE-aware manner, we use the Hierarchical Token Bucket Filter (HTB) [14]. Its tree structure allows to define a hierarchy for borrowing between leaf nodes (QoS Flows) with a common parent (slices) and thus, HTB can be used for the following evaluation cases: QoS Flows only, slices only, as well as the combination of both concepts. It allows us to achieve both, the desired properties of inter-slice resource isolation and intra-slice resource sharing. To this end, the configuration includes GBR and MBR settings per slice (inner nodes/parent) as well as per flow within a slice (leaves). At this point, we want to emphasize that we do not propose to use HTB in a real 5G system. To us, it only serves as an efficient solution within the OMNeT++ simulator. How resource slicing or enforcing GBR/MBR is actually achieved depends on how RAN vendors implement their own RAN scheduler [15]–[17].

IV. Methodology

This section introduces our evaluation methodology. We first describe the used simulation setup and the included applications along with their utility functions. We then detail on our implementation of HTB and our abstraction of the network slicing approach.

A. OMNeT++ Simulation Environment

Our setup is illustrated in Figure 1 and based on the OMNeT++ network simulator [18] along with the INET framework.[3] A single link connects the client network with the server network, which hosts a server for each investigated application type. We use standard datarate channels provided by OMNeT++ and model the bitrate and delay of a generic wired connection with a bit error rate of 0. The links are connected to the devices using point-to-point (PPP) interfaces provided by the INET framework. They use a standard drop tail queue on the outbound side of the interface with a non-blocking inbound. Additionally, the queue on the interfaces between the two routers is coupled with an HTB module used for QoS guarantees as described in IV-D.

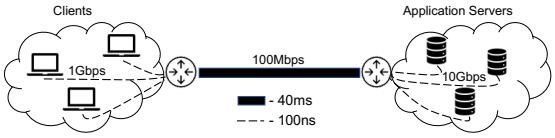


Fig. 1: Simulation setup

B. Application Mix

Our setup includes five different applications: DASH-based Video on Demand Streaming (*VoD*), DASH-based Live Video Streaming (*Live*), File Download (*FD*), Secure Shell (*SSH*), and Voice over IP (*VoIP*). These applications differ w.r.t. their requirements in terms of bandwidth and delay. While *VoD* and *FD* are bandwidth-hungry applications that can cope with a certain delay, the quality of *SSH* and *VoIP* is mainly determined by the delay. For *Live*, both, the available bandwidth and the delay, are relevant factors.

For *VoIP*, we use an implementation supplied with the INET framework. It is modelled as one-way communication of two parties, one being a listener and another being a talker (receiver and sender). The sender has on/off talkspurt behavior, with some silent intervals when no packets are being sent. The size of the VoIP packets is 40B each and a new packet is generated every 20ms. The receiving buffer has a size of 20 packets.

SSH is used for secure remote access operation. Our SSH client is based on the Telnet application provided by the INET framework. The application mimics a behavior of a user typing a remote control command with 50ms per keystroke, and then waiting for the output of 500B. To account for encryption overhead (*SHA1*), we increase the packet size for key presses to 40B. One second after the output of the command is received, the user types another command and does so repeatedly in a loop.

The *FD* client is based on the basic TCP client provided by the INET framework. Via a request of 800B length, it downloads a file of 10MiB. After receiving the file, the client waits 1s before initiating the next file download.

The *VoD* client is based on an existing implementation of an adaptive TCP-based video streaming [4] The server provides videos in small segments of five seconds, each encoded with 4 different quality levels. Depending on the buffer state, the client dynamically selects the quality for the next segment with switches occuring at buffer states of 10, 20, and 30s. The maximum buffer capacity is 40 seconds. The size of the manifest file containing the video specifications is 100kB and the segment request size is 200B. The overall duration of the video follows a uniform distribution in the range of (240,320)s.

For the *Live* client, we adapt the VoD client's implementation so as to meet the characteristics of Live streaming. Instead of a buffer-based quality adaptation, the client selects

[3]https://inet.omnetpp.org/Introduction.html. Accessed 09.06.2021.

[4]https://github.com/andersonandrei/adaptive-video-tcp-omnet. Accessed 31.07.2021.

the quality based on throughput estimations from previous segment downloads. The segment duration is decreased to 1s, and the maximum buffer capacity is set to 6s. If the delay to the live edge[5] exceeds 4 seconds, the playback speed is increased by 5% to catch up. Due to the specifics of its implementation, a video duration needs to be set for the Live client. It is selected from the range between (40,70)s.

C. Utility Functions

In order to allocate resources to flows and slices in a QoE-aware manner, we need to know how delay and available throughput - the metrics which are at least to a certain extent controllable - affect the different applications' QoE. Hence, to quantify the application-specific network requirements, we make use of *utility functions* which map different delay/throughput combinations to QoE. We prepare these utility functions by running experiments that cover broad delay- and throughput-settings, in which we monitor the network- and application-related KPIs, and apply existing QoE models (which map the KPIs to QoE) so to obtain the resulting MOS values.

In the following, we briefly present the QoE models used for the five applications considered in this paper. For *VoIP*, we use the ITU-T E-model [19]. The key factors considered by the model are: mouth-to-ear delay, loss rate, signal-to-noise ratio, and equipment impairment factor. For *VoD* and *Live* the standardized ITU-T P.1203 model is applied [20]. As input KPIs it considers the initial delay, number and duration of video interruptions, the delivered video quality and its fluctuation. The *SSH* QoE model is based on the QoE model for Remote Virtual Desktop Services (RVDS) from [21]. According to the approach originally introduced in [2], a fitting function is used to adjust the model for our scenarios. The QoE model for *SSH* uses the round-trip-time (RTT) as the only KPI. Finally, we use the model from [22] which defines the MOS for the *FD* application as a logarithmic function of overall download time.

The resulting MOS values from these models do not always correspond to the full MOS range of $[1.0, 5.0]$ since they depend on the QoE model. The results are illustrated as heatmaps in Figure 2, where the bandwidth capacity is denoted on the x-axis and the delay on the y-axis. As not any combination of bandwidth and delay is possible, i.e. very small bandwidth settings may lead to increased delays, we additionally evaluate for each application its bandwidth to delay relation, to determine on a per-application scale the minimum bandwidth capacity to achieve a certain QoE. This relation is shown as a red curve in Figure 2. The white dashed line denotes the bitrate which corresponds to a QoE of 3.5.

Figure 2a shows that MOS of *VoD* increases rapidly with increasing bandwidth and reaches a MOS of 4.4. The delay has little impact on the QoE until an end-to-end delay of 250ms is exceeded. *Live*, as shown in Figure 2b, reaches a maximum QoE of 4.58. Its MOS drops quickly with increasing end-to-end delay. *SSH* and *VoIP* applications shown in Figure 2c and 2d, behave similarly with their MOS linearly decreasing with growing delay. Whereas *SSH* can achieve a high QoE with a very limited bandwidth of 5 kbps, the *VoIP* client shows an on-off behavior, requiring at least 30 kbps to operate during the on-phases. The MOS of *FD* increases linearly with increasing bandwidth until a MOS score of 4.72.

[5]The moment at which live video segment is generated.

D. Traffic Control Mechanism: Hierarchical Token Bucket

In order to enable co-existence of numerous heterogeneous applications, 5G networks require efficient rate control mechanisms. According to the 3GPP standards for the Next Generation Radio Access Networks (NG-RAN) in 5G [12], each flow should have its assigned *guaranteed bitrate (GBR)* and *maximum bitrate (MBR)*.

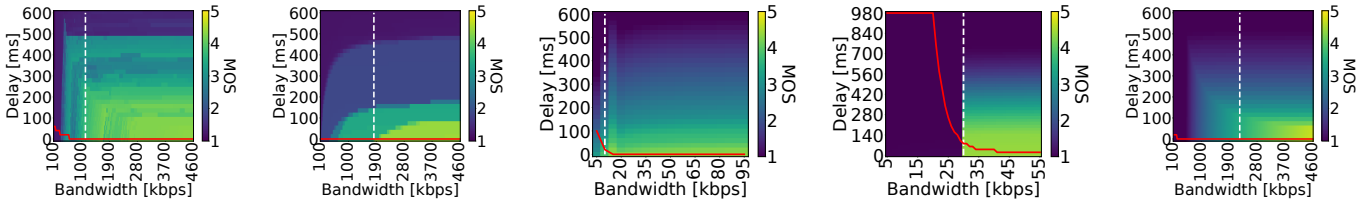
There is no simple ready-to-use OMNeT++ module that emulates network slicing and at the same time supports such two-level bitrate guarantees. Therefore, we implemented a new OMNeT++ compound module called *HTBQueue* that adopts the classful queuing approach - Hierarchical Token Bucket (HTB). We decided to emulate slicing with HTB for several reasons. Firstly, with the HTB tree-based structure we introduce the hierarchy needed for the realization of slices, as shown in [23]. Secondly, HTB has a high rate conformance [24]. HTB in general is costly in terms of CPU/memory usage [23], [25], especially when it comes to large networks with numerous flows. However, due to the simulative nature of our approach, scalability is not an issue for our evaluations carried out in this work.

The *HTBQueue* is based on the Linux HTB implementation [14]. The key structure is the HTB tree, an example is shown in Figure 3. The tree consists of three different types of nodes/classes: root, inner, and leaf. All three HTB classes have the following parameters: assured rate (equivalent to GBR) and ceiling rate (maximum achievable rate, equivalent to MBR). The root's assured and ceiling rate are set to the link bandwidth. A key constraint in HTB is that the sum of the assured rates of one node's children has to be less than or equal to the assured rate of that node. In general, only the leaves have queues and priorities assigned to them.

We tested and validated that our implementation of HTB in OMNeT++ behaves as expected. In our evaluations we compared the *HTBQueue* performance with the expected outcomes of the same scenarios in Linux. We also made sure that HTB and TCP are interacting in the correct way in terms of analysing the throughput values, RTT, and inter-departure times as well as congestion window and queue sizes.

E. Network Slicing Abstraction and Assumptions

For our evaluations, we do not actually apply network slicing, but mimic its behavior via HTB. This allows to focus on the impact of data plane traffic engineering and we omit any control plane involvement or function deployment which happens in the background. Nevertheless, our proposed approach is still realistic in terms of the data plane traffic control capabilities, as all necessary functionalities can be provided via 5G NFs. This includes for example the propagation of QoE and



(a) VoD (b) Live (c) SSH (d) VoIP (e) FD

Fig. 2: Utility functions of the different applications to quantify resulting QoE for different bandwidth/delay combinations. Red line shows applications' bandwidth-delay relationship. White dashed line denotes a bitrate corresponding to QoE of 3.5.

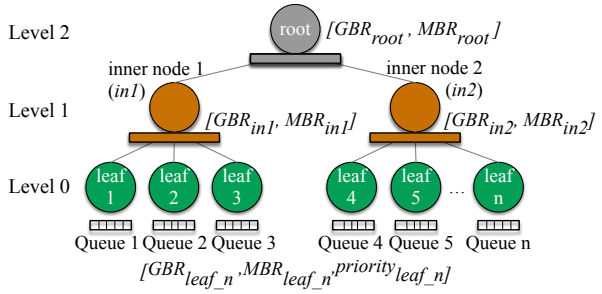


Fig. 3: An exemplary tree structure for HTB.

user-centric information via the Application Function (AF), which is needed by network operators to tune their network to accommodate the needs of heterogeneous applications.

However, we want to emphasize that our evaluations are based on per-flow QoS control using an adaptation of HTB for OMNeT++. For simplification, we assume one active flow per client. QoS guarantees are achieved by assigning one client to each HTB leaf that represents a single QoS Flow. The slices are emulated either by placing all flows of the same application in a standalone, bandwidth-limited network or using the inner-classes of HTB. This means that we are considering a scenario without any encapsulation overhead associated with virtualization. Furthermore, the number of flows is pre-configured and static during the simulation.

V. Evaluation

In the following, we describe our experiment settings and the conducted experiments. Afterwards, we present our results and shortly describe the implications and limitations of our study.

A. Simulation Parameters and Settings

This sections describes the simulation context, the parameter settings, and evaluation metrics used for our experiments.

1) Per-Application QoS Profiles: We assume that the QoS profiles are fixed up-front for each application. The GBRs are set so as to achieve a QoE of 3.5, i.e., an acceptable user satisfaction, according to the utility functions described in IV-C. This results in the following per-application QoS-profiles, referred to as $GBR^a_{original}$ for application a: 30 kbps for *VoIP*, 10 kbps for *SSH*, 1120 kbps for *VoD*, 1820 kbps for *Live*, and 2240 kbps for *FD*.

2) Application Mix, Slice Dimensioning, and Admission Control: We consider the following distribution for the clients' used applications, expressed as fraction f_a for a given application a: $f_{VoD} = 40\%$, $f_{Live} = 20\%$, $f_{FD} = 5\%$, $f_{VoIP} = 30\%$, and $f_{SSH} = 5\%$. The traffic mix roughly mimics the statistics for mobile data volume collected in January 2021[6]. In the scenarios which include slicing, we dimension the slices up-front according to the expected applications' probabilities and QoS profiles. We calculate a multiplier m based on a sum of QoS profile bitrates $GBR^a_{original}$ of all applications a weighted according to the traffic mix f_a and the available system capacity c:

$$m = \frac{c}{\sum_a f_a \cdot GBR^a_{original}} \tag{1}$$

The multiplier is then used to calculate the slice capacity c^a_s for each application type a:

$$c^a_s = m \cdot f_a \cdot GBR^a_{original} \tag{2}$$

The admission control is decoupled form the simulation itself. We assume that 150 clients try to enter the system. For each of these clients, we randomly determine its application type, following the traffic mix probabilities described above. The client is admitted if its QoS profile can be served, considering the remaining system capacity (or slice capacity) in terms of bandwidth.

3) Evaluation Metrics: We compare the performance of the different scenarios based on the following metrics:

Number of admitted clients: The more clients can be admitted the better. Besides the absolute number of admitted clients, we also consider the rejection rates, i.e., the share of rejected clients from the 150 clients that try to enter the system.

Network resource utilization: Optimally, the available network resources are fully utilized. To study in how far this goal can be achieved in the different scenarios, we compare them according to their relative usage of the overall system resources. For scenarios that involve slices, we additionally consider the resource utilization within a specific slice.

MOS: Finally, we compare the different scenarios with respect to their performance in terms of QoE, expressed on MOS scale. Higher values represent a better user satisfaction.

[6]Taken from https://www.statista.com/statistics/383715/global-mobile-data-traffic-share/. Accessed 31.07.2021.

TABLE I: Overview of evaluation scenarios.

Scenario	Slicing	Flow control	GBR	MBR
BE-1/BE-2	No	No flow control, best effort	Not specified	Not specified
QoS	No	QoS-based	$GBR_{original}$	$\{1, 1.25, 1.5, 1.75, 2\}$ multiplier of the $GBR_{original}$
QoS-Sli-1	Yes, full isolation	Same as in QoS	Same as in QoS	Same as in QoS
QoS-Sli-2	Yes, via HTB	Same as in QoS	Same as in QoS	Same as in QoS
QoE	No	Based on the Parameter Study	See Table II	See Table II
QoE-Sli-1	Yes, full isolation	Same as in QoE	Same as in QoE	Same as in QoE
QoE-Sli-2	Yes, via HTB	Same as in QoE	Same as in QoE	Same as in QoE

B. Experiment Descriptions

1) Parameter Study on GBR/MBR Settings: In order to tune the per-flow GBR/MBR settings within slices so as to increase the number of admitted clients whilst preserving the obtained QoE, we perform a parameter study for each of the five application types. We start with GBR settings $GBR^a_{original}$ according to the QoS profiles described above. The initial number of clients in the system is set to $n = 100$. We increase the number of admitted clients step by step and decrease the GBRs from the QoS profiles using the following set of scaling factors $s \in \{0.95, 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, 0.5\}$. The corresponding number of clients admitted after lowering $GBR^a_{original}$ is obtained as $\frac{n}{s}$, resulting in numbers of clients $n \in \{111, 117, 125, 133, 142, 153, 166, 181, 200\}$. The MBRs are set as the $\{1, 1.25, 1.5, 1.75, 2\}$-fold of $GBR^a_{original}$. The aim is to study the actual (from a QoE perspective) required GBRs for the different applications.

2) Evaluation Scenarios: We evaluate different scenarios which differ with respect to their QoS settings and the applied control mechanism(s). While Table I gives a brief overview, the scenarios are described in more detail in the following.

Best-effort Baseline (BE): In this scenario, we do not grant any bitrate guarantees to flows. The flows behave in a best-effort manner and no explicit control mechanism is applied. First, we consider scenario BE-1 where we admit all 150 incoming clients. In order to be comparable with the other scenarios, we consider in scenario BE-2 the same number of clients per application as in the QoS scenario.

QoS Flows (QoS): This scenario considers both GBRs and MBRs on a per-flow level without slices. While the MBRs are set as the $\{1, 1.25, 1.5, 1.75, 2\}$-fold multiple of the GBR, the GBR are set according to the QoS profiles described in V-A1. In this scenario, each flow can borrow from any other flow.

QoS Flows Combined with Slices (QoS-Sli): Same as in the QoS scenario, we consider the predefined GBRs and MBRs on a per-flow level, but additionally introduce five slices, each representing one application type. They are dimensioned as follows: 47.991 Mbps for the *VoD* slice, 38.993 Mbps for the *Live* slice, 11.997 Mbps for the *FD* slice, 0.964 Mbps for the *VoIP* slice, and 0.053 Mbps for the *SSH* slice. In this scenario, a flow can only borrow from flows in the same slice, i.e. we do not allow borrowing between slices due to the isolation of slices. We mimic the slicing behavior in two ways: (1) Assuming a perfect isolation (QoS-Sli-1). In this case, we run a dedicated simulation for each slice with the specified per flow and per slice settings, to eliminate any potential interference between different applications' flows. (2) Assuming effects due to the shared environment (QoS-Sli-2). In this case, all slices are run simultaneously in one simulation.

QoE Flows (QoE): Different from the QoS scenario, we do not rely on the pre-defined GBRs. Instead, we set the GBRs and MBRs for the different application types according to what we have found as an improved setting during the parameter study, as shown in Table II.

QoE-aware Slicing (QoE-Sli): Similar to QoS-Sli scenario, we use slicing but combine it with GBR and MBR setting for each application type taken from the QoE scenario. This allows us to admit more clients compared to QoS-Sli, simultaneously keeping the guaranteed QoE of each application. Same as in QoS-Sli, we mimic the slicing behavior once assuming perfect isolation (QoE-Sli-1) and once assuming possible interference between the slices (QoE-Sli-2). The resources allocated to each slice are the same as in QoS-Sli.

C. Evaluation Results

In the following, we first summarize the results from the conducted parameter study and afterwards detail on the obtained MOS and resource utilization.

1) Parameter study on GBR/MBR settings: Figure 4 shows the results from the parameter study, which lead to the adapted per-flow MBR/GBR settings. For reasons of clarity, we only show an extract of the whole parameter study, namely those experiments, where the number of clients corresponds to the maximum number of clients that can be served for a given GBR setting under the overall system capacity constraint[7]. Hence, we show one out of eleven settings for the number of active clients. The GBR setting, expressed as the percentage compared to the QoS Flows' GBR capacity, is noted on the x-axis. The y-axis shows the MBR multiplier. While *SSH* is neither sensitive to the GBR-, nor to the MBR-setting, the MOS of *VoIP* is only affected by the GBR. The QoE of *VoD*, *Live*, and *FD* is affected by both rate settings, but more considerably by the GBR configuration. We choose the optimized setting so that the GBR is as low as possible while still obtaining a MOS of at least 3.5. The chosen values are marked in red and listed in Table II.

The *VoD* clients' on-off behaviour allows for high MOS, until a GBR of roughly 85% where the available bandwidth can still be efficiently distributed between clients. *Live* behaves similarly and shows a linear relationship between MOS and GBR. However, it is more sensitive to increasing MBR, due to the ability of estimating higher bitrate that allows for downloading higher quality segments. The MOS of *SSH* is not affected by the changing GBR and MBR settings. For *VoIP*, we observe a high MOS until a clear cutoff point at 65% GBR setting. Until then, the traffic can be distributed so that all clients can obtain the required 30 kbps during the on-period. The MOS of *FD* relies solely on the available bandwidth and accordingly only on the GBR setting.

[7]The sum of all clients' GBRs cannot exceed the overall system capacity.

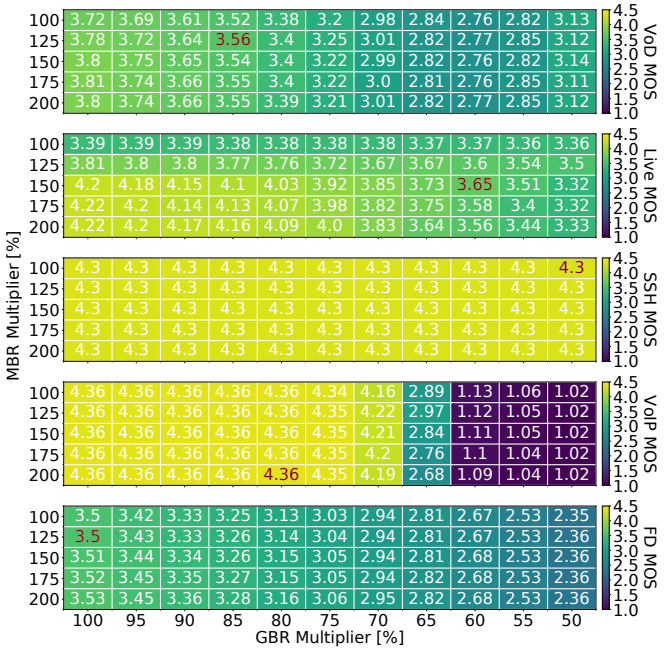


Fig. 4: Average MOS according to GBR and MBR multipliers for each application type. Values in red indicate which multipliers were chosen for the QoE scenarios.

TABLE II: Suggested QoE-aware GBR/MBR settings.

Application a	$GBR^a_{optimal}$ ($\%$ $GBR^a_{original}$)	$GBR^a_{optimal}$ $[kbps]$	$MBR^a_{optimal}$ ($\%$ $GBR^a_{original}$)	$MBR^a_{optimal}$ $[kbps]$
VoD	85	952	125	1400
Live	60	1092	150	2730
FD	100	2240	125	2800
VoIP	80	24	200	60
SSH	50	5	100	10

2) Performance Evaluation of the Different Scenarios: Figure 5 depicts the average MOS and the number of admitted clients. The number of admitted clients per application type and corresponding rejection rates are presented in Table III. In both best-effort scenarios (BE-1 and BE-2), the average MOS of *Live* and *FD* clients are below the targeted QoE guarantee of 3.5. For any MBR setting, the QoS scenario outperforms BE-2 in terms of the overall average MOS (denoted as black dots), whilst admitting the same number of clients. While BE-2 results in an average MOS of 3.57, this can be increased to up to 4.14 by the QoS scenario with an MBR set to 150% of the GBR. We furthermore see that the average MOS values of the different application types, and hence the fulfillment of the QoE target, is sensitive to the MBR setting. The QoE target is reached for all applications in all QoE scenarios except for *Live* and *FD* in cases where MBR is set to 100% of GBR.

As soon as we introduce one slice per application additionally to the QoS Flows (QoS-Sli-1), the number of admitted clients is reduced to 105 with the total rejection rate increasing to 30% and the sensitivity towards the MBR setting is reduced. Indeed, we see some improvement in terms of MOS with increasing MBRs, but these effects become negligible as soon as the MBR reaches a value of 150% of the GBR. The application most profiting from the introduction of slicing is *Live*. In the QoS-Sli-1 scenario with MBR being 150% of GBR, its average MOS can be increased by up to 0.46 compared to the corresponding QoS-scenario. However, when we do not assume perfect isolation between the slices (QoS-Sli-2), we observe a slight decrease of the MOS, most dominantly for *Live* and *FD*. This can be explained with an interplay between extra delay introduced with HTB and TCP, having an effect on *Live* as a delay sensitive application and on download time which is the most important KPI for *FD*.

TABLE III: Number of admitted clients for each scenario. The number of incoming clients is set to 150 in all scenarios.

Scenario	Admitted clients					Rejection rate (%)					Total	Total reje.
	VoD	*Live*	*FD*	*VoIP*	*SSH*	*VoD*	*Live*	*FD*	*VoIP*	*SSH*	admitted	rate (%)
BE-1	64	28	8	40	10	0	0	0	0	0	150	0
BE-2	47	19	5	40	10	26.6	32.1	37.5	0	0	121	19.3
QoS	47	19	5	40	10	26.6	32.1	37.5	0	0	121	19.3
QoS-Sli-1	42	21	5	32	5	34.4	25	37.5	20	50	105	30
QoS-Sli-2	42	21	5	32	5	34.4	25	37.5	20	50	105	30
QoE	57	26	7	40	10	10.9	7.1	12.5	0	0	140	6.6
QoE-Sli-1	50	28	5	40	10	21.9	0	37.5	0	0	133	11.3
QoE-Sli-2	50	28	5	40	10	21.9	0	37.5	0	0	133	11.3

Finally, we discuss the results for the QoE-aware scenarios, i.e., with GBR/MBR settings tuned according to the outcomes of the parameter study. Using these optimized settings without slicing (QoE) allows to admit 140 clients (mostly due to lower rejection rate of *VoD* and *Live* clients) and hence more than in any of the QoS-aware scenarios. However, for *Live* the QoE target is barely met. This can be overcome by combining the optimized GBR and MBR settings with slicing. When assuming perfect isolation (QoE-Sli-1), the average MOS of all applications exceed the target value of 3.5 and the overall MOS is 3.93. If the resources cannot be fully isolated (QoE-Sli-2), the overall average reduces to 3.72. But still, the QoE target can be met for each of the applications. Due to the loss of complexity gain, the introduction of slicing (QoE-Sli-1,QoE-Sli-2) reduces the number of admitted clients to 133, compared to the scenario without slices (QoE) where 140 clients could be admitted. However, we want to emphasize that it is still higher than in any of the QoS scenarios (QoS, QoS-Sli-1, QoS-Sli-2). Hence, with the proposed settings, we are capable to guarantee a given QoE level, whilst admitting more clients.

Next, we study the resource utilization using Figure 6. In the best effort cases (BE-1 and BE-2), where the flows are not constrained by QoS control, the system is utilized to 100%. In the QoS scenario, only 71% of the resources are exploited if the clients are not allowed to borrow, i.e., the MBR is set as 100%. The utilization increases with increasing MBR setting. A borrowing rate of 200% results in an overall system utilization of 99.8%. Introducing slices and assuming perfect isolation (QoS-Sli-1) slightly reduces the overall resource utilization. Despite the introduced constraints in terms of borrowing (only flows from flows of the same slice), the

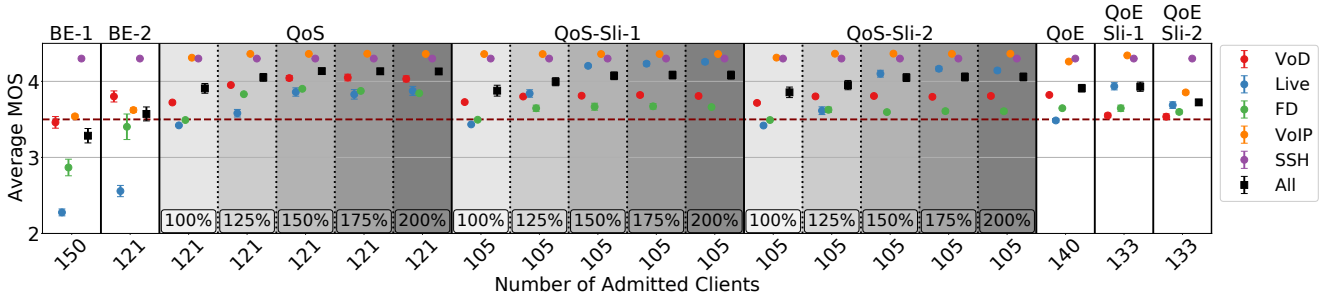


Fig. 5: MOS and number of admitted clients. The horizontal line represents the targeted QoE guarantee of 3.5. Dots denote average MOS, errorbars the 95% confidence interval, percentages in boxes the selected MBR multiplier in QoS and QoS-Sli.

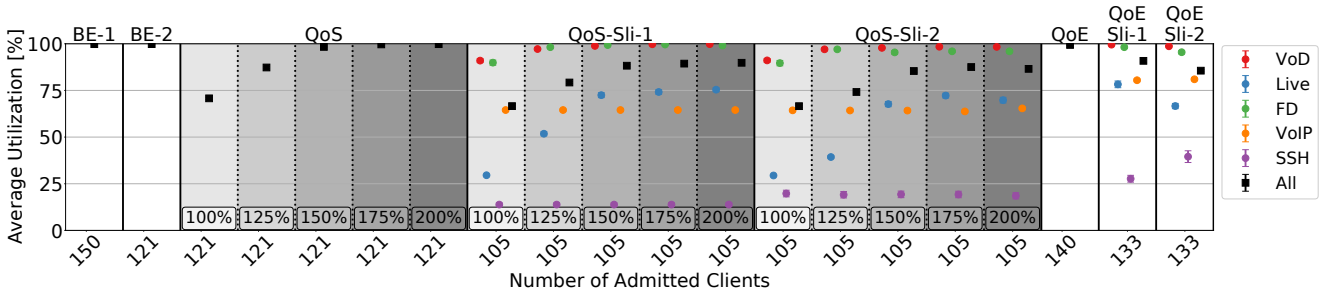


Fig. 6: System/Slice utilization and number of admitted clients for different scenarios. Dots denote average the system utilization, errorbars the 95% confidence interval, percentages in boxes the selected MBR multiplier in QoS and QoS-Sli scenarios.

reduction of utilization is in any case below an absolute value of 10%, which is observed for an MBR setting of 150%. While the inner-slice utilization is relatively static for *VoIP* and *SSH*, the utilization of the other slices can be increased with an increasing MBR setting up to 175%. This effect is most dominant for *Live*. The results obtained when not assuming perfect isolation (QoS-Sli-2) hardly differ from QoS-Sli-1. As expected, the overall utilization is slightly lower, mainly due to the reduced obtained throughput within *VoD*, *Live*, and *FD* slices. The slice utilization increased for *SSH*.

Applying the optimized settings in the QoE scenario results in a similar system utilization as QoS with an MBR setting of 150%. For the QoE-Sli-1 scenario, the overall system utilization is 90.85% percent, and thus higher than in any of the QoS-Sli-X scenarios. Please note that the number of admitted clients is increased by 26.7%, whilst simultaneously keeping a comparable average QoE score. When not assuming perfect isolation (QoE-Sli-2), the overall utilization slightly decreases to a value of 85.59%, which can be explained by the reduced utilization of the *Live* slice.

The system is not fully utilized in the QoE-Sli-X scenarios, which indicates that the GBR settings can be even more constrained, allowing to admit more clients. This trade-off between admitted clients and the ability to still meet the QoE target, however, needs to be studied in detail, which is left for future work.

D. Limitations of the Study

Our evaluations are based on several abstractions and assumptions. First, we rely on a packet simulator and we omit the network's radio part. However, our goal was to study the impact of applying QoS Flows and network slices as well as the implications of combining them and tuning their settings, i.e., per-flow MBRs and GBRs and slice-specific resources. These parameters can also be specified end-to-end in real networks. Therefore, RAN and core network QoS settings need to be geared to each other, so as to realize actual end-to-end slicing and QoS flow settings. In particular, our methodology is not limited to the current simulation setup, but can be applied similarly in a real mobile network.

Additionally, our studies are confined to five types of applications and we assumed one specific client distribution for them. The generalizability is hence limited due to this specific traffic mix. However, our set of services is representative in terms of their sensitivity to delay and throughput: while *VoD* and *FD* are bandwidth-hungry applications, *VoIP* and *SSH* have high delay requirements, and finally *Live* is sensitive to both. Furthermore, the proposed methodology can be applied to any other type of application without restrictions.

Besides the given applications and their probabilities, the evaluated parameter space is limited in terms of the number of clients, the overall capacities, and the chosen QoE target of 3.5. Nevertheless, we expect that the revealed benefits of our proposed approach hold in general and that the qualitative relationship will hold except for differences in terms of absolute values. Finally, we confined on HTB to mimic the scheduler's behavior and focused on bandwidth-related parameters when assigning QoS resources, while considering the delay only indirectly via the utility functions. Setting GBR and MBR

values is only a subset of the capabilities introduced with QoS Flows. Broadening the experiments by means of using different scheduling mechanisms and by better exploiting the capabilities of QoS Flows is left for future work.

VI. Conclusion

This work evaluated the potential of resource allocation using QoS Flows and network slicing as introduced in the 5G architecture. Particularly, we focus on system operation, i.e., how to dimension the slices and properly set per-flow bitrates with respect to application demands and resource efficiency. In an extensive simulation-based parameter study, we tuned system parameters to obtain a targeted MOS for all application types. As our next step, we combined the capabilities of per-flow bitrate settings and network slicing and investigated different degrees of isolation. Our results show the potentials of a fine-tuned, application-aware setting of per-flow and slice parameters. We show that the number of admitted clients can be increased, whilst being able to provide a given target QoE score to all active applications. This can be achieved by better exploiting the available resource, i.e., an increase of the overall system utilization.

Future work needs to further investigate the trade-off between system load, i.e., the number of admitted clients, and the ability to satisfy given QoE targets, and the impact of application grouping into slices. Furthermore, while so far relying on intra-slice borrowing only, more complex inter-slice borrowing strategies, e.g., by introducing prioritized and non-prioritized per-application slices need to be investigated.

Acknowledgment

This work has been funded by the German BMBF Software Campus Grant "BigQoE" (01IS17052) and was supported by EC H2020 TeraFlow (101015857). The authors thank Martin Devera, who implemented the Linux HTB, for his continuous support and guidance while developing the HTB OMNeT++ module.

References

[1] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on selected areas in communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[2] C. Sieber, S. Schwarzmann, A. Blenk, T. Zinner, and W. Kellerer, "Scalable application-and user-aware resource allocation in enterprise networks using end-host pacing," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 5, no. 3, pp. 1–41, 2020.

[3] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE journal on selected areas in communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[4] Z. Kotulski, T. Nowak, M. Sepczuk, M. Tunia, R. Artych, K. Bocianiak, T. Osko, and J.-P. Wary, "On end-to-end approach for slice isolation in 5G networks. fundamental challenges," in *Federated conference on computer science and information systems (FedCSIS)*. IEEE, 2017, pp. 783–792.

[5] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[6] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, 2017.

[7] Q. Wang, J. Alcaraz-Calero, R. Ricart-Sanchez, M. B. Weiss, A. Gavras, N. Nikaein, X. Vasilakos, B. Giacomo, G. Pietro, M. Roddy *et al.*, "Enable advanced QoS-aware network slicing in 5G networks for slice-based media use cases," *IEEE Transactions on Broadcasting*, vol. 65, no. 2, pp. 444–453, 2019.

[8] M. Gramaglia, I. Digon, V. Friderikos, D. von Hugo, C. Mannweiler, M. A. Puente, K. Samdanis, and B. Sayadi, "Flexible connectivity and QoE/QoS management for 5G networks: The 5G NORMA view," in *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016, pp. 373–379.

[9] F. Z. Yousaf, M. Gramaglia, V. Friderikos, B. Gajic, D. von Hugo, B. Sayadi, V. Sciancalepore, and M. R. Crippa, "Network slicing with flexible mobility and QoS/QoE support for 5G networks," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2017, pp. 1195–1201.

[10] J. Kim and M. Xie, "A study of slice-aware service assurance for network function virtualization," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 489–497.

[11] 3rd Generation Partnership Project (3GPP), "System architecture for the 5G System (5GS)," Technical Specification (TS) 23.501, Mar. 2021, version 17.0.0.

[12] ——, "NR; NR and NG-RAN Overall description; Stage-2," Technical Specification (TS) 38.300, Mar. 2021, release 16.5.0.

[13] W. da Silva Coelho, A. Benhamiche, N. Perrot, and S. Secci, "On the impact of novel function mappings, sharing policies, and split settings in network slice design," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–9.

[14] M. Devera, "Hierarchical token bucket," http://luxik.cdi.cz/~devik/qos/htb/, February 2002, [Online; accessed 16-March-2021].

[15] F. Wamser, S. Deschner, T. Zinner, and P. Tran-Gia, "Investigation of different approaches for qoe-oriented scheduling in OFDMA networks," in *International Conference on Mobile Networks and Management*. Springer, 2013, pp. 172–187.

[16] D. Fernàndez and H. Montes, "An enhanced quality of service method for guaranteed bitrate services over shared channels in egprs systems," in *Vehicular Technology Conference. IEEE 55th Vehicular Technology Conference. VTC Spring 2002*, vol. 2. IEEE, 2002, pp. 957–961.

[17] F. Wamser, D. Staehle, J. Prokopec, A. Maeder, and P. Tran-Gia, "Utilizing buffered youtube playtime for qoe-oriented scheduling in ofdma networks," in *24th International Teletraffic Congress (ITC 24)*. IEEE, 2012, pp. 1–8.

[18] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," 01 2008, p. 60.

[19] I. Rec, "G. 107 - The e model, a computational model for use in transmission planning," *International Telecommunication Union*, vol. 8, no. 20, 2003.

[20] A. Raake, M.-N. Garcia, W. Robitza, P. List, S. Göring, and B. Feiten, "A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P. 1203.1," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2017, pp. 1–6.

[21] P. Casas, M. Seufert, S. Egger, and R. Schatz, "Quality of experience in remote virtual desktop services," in *IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2013, pp. 1352–1357.

[22] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz, ""Time is bandwidth"? Narrowing the gap between subjective time perception and Quality of Experience," in *2012 IEEE international conference on communications (ICC)*. IEEE, 2012, pp. 1325–1330.

[23] A. Saeed, N. Dukkipati, V. Valancius, T. Lam, C. Contavalli, and A. Vahdat, "Carousel: Scalable traffic shaping at end-hosts," in *ACM SIGCOMM*, 2017.

[24] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelém, "Control of multiple packet schedulers for improving qos on openflow/sdn networking," in *2nd European Workshop on Software Defined Networks*, 2013, pp. 81–86.

[25] S. Radhakrishnan, Y. Geng, V. Jeyakumar, A. Kabbani, G. Porter, and A. Vahdat, "SENIC: Scalable NIC for end-host rate limiting," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 475–488.