Fair Resource Sharing for Stateless-Core Packet-Switched Networks with Prioritization

Michael Menth and Nikolas Zeitler
University of Tuebingen, Chair of Communication Networks, Germany

***Abstract*—We propose activity-based congestion management (ABC) to enforce fair bandwidth sharing among users in packet-based communication networks without requiring per-user information in forwarding nodes. Activity relates to the sent data rate of a user and its contracted reference rate. Activity meters monitor user traffic at edge nodes and add activity information to packets. Forwarding nodes utilize this information to preferentially drop packets with high activity values in case of congestion.**

In this paper we significantly simplify the behavior of ABC's control devices: the activity meter in edge nodes and the activity AQM in forwarding nodes. Moreover, we introduce scheduling priorities for ABC, provide means to avoid starvation of non-prioritized traffic through prioritized traffic, and means to disincentivize prioritized transmission of non-realtime traffic.

We study the fairness achieved with and without ABC using packet-based simulation, mostly under challenging conditions where a heavy user wants to monopolize a bottleneck's bandwidth. We investigate bandwidth sharing for non-responsive traffic, for responsive traffic, and a mixture of both. The effect of traffic prioritization is studied. We explore the impact of configuration parameters and recommend meaningful system parameters. We illustrate their impact on system dynamics and show that ABC expedites sporadic uploads, which improves the quality of experience for interactive applications. Finally, we compare the performance of ABC with the one of CSFQ whose objective is the same as the one of ABC, but significantly differs with regard to signalled information and algorithmic approach.

Summarizing, ABC creates an ecosystem where users can maximize their throughput if they do not exceed their fair share on the bottleneck link. This incentivizes the usage of congestion-controlled transport protocols. Moreover, ABC protects traffic from users applying congestion control against traffic overload from users not leveraging congestion control, even if the latter send prioritized traffic.

***Index Terms*—Packet-switched networks, congestion management, fairness, TCP**

I. Introduction

Future mobile networks will consist of many small cells, each of them capable to provide large bandwidths to users [1], [2]. This makes appropriate overprovisioning of meshed backhaul networks diffcult and costly so that means are needed for congestion management in case of overload. Similar problems arise in multi-tenant datacenter networks with lots of applications generating unknown traffic patterns and overprovisioned core switches or in Internet service providers' residential access networks. In such scenarios, scalable and quality of service aware bandwidth sharing is desired in case of congestion. That means, network nodes should remain simple, not require per-user contexts, and provide means for service differentiation.

We consider congestion management that detects congestion in a network and takes actions to resolve it. Heavy users may transmit unresponsive flows or lots of congestion-controlled flows so that they achieve significantly more throughput than light users in the absence of fairness-aware congestion management. The latter requires that appropriate packets are dropped to protect the traffic of light users against overload created by heavy users. Active queue management (AQM) is a simple congestion management approach. However, without additional information, AQM reduces queue length and queueing time, but it cannot prevent heavy users from monopolizing a bottleneck link's bandwidth. This significantly compromises the quality of experience of light users. More sophisticated congestion management variants guarantee some fairness among users. Some ISPs apply back pressure on certain users, traffic, or applications [3]–[5]. Thereby, better service can be delivered to most customers and reinvestments in transmission capacity can be delayed, resulting in cost savings. Scheduling mechanisms like Weighted Fair Queuing (WFQ) and its variants are standard methods to enforce fairness in packet-based networks, but they need to associate packets with users, which requires per-user states and signalling in core nodes and introduces complexity.

In the IETF, several efforts have been taken in the recent years to better manage congestion in the Internet. Congestion control for real-time traffic sent over RTP (Real-Time Protocol) have been proposed in the RMCAT (RTP Media Congestion Avoidance Techniques) working group [6]. LEDBAT (Low Extra Delay Background Transport) [7] has been standardized for transport of background traffic to give way for normal traffic in case of congestion. Recently, novel AQM mechanisms have been described to cope with variable-capacity links [8], [9] as they occur in wireless networks. Most notably, congestion exposure (ConEx) was introduced to expose congestion caused by a flow in its IP header to provide more information for congestion management to improve fairness among users [10]. One of its goals was to incentivize the application of congestion-sensitive protocols like LEDBAT for background traffic. Mobile access networks [11], residential access networks [12], and datacenters [13] were considered as use cases.

The authors acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG) under grant ME2727/2-1. The authors alone are responsible for the content of the paper.

In [14] we proposed activity-based congestion management (ABC). Activity meters at the network edge measure the users' activity and tag it to their packets. Forwarding nodes use that information to preferentially drop traffic from more active users in case of congestion. With ABC control, users can maximize their throughput by sending traffic at their fair share with regard to the capacity of the bottleneck link. Thus, ABC creates an eco-system giving incentives to users to apply congestion control. ABC was essentially developed because ConEx-based congestion management turned out to be little efficient in simulation studies. Therefore, ABC may be applied in the same use cases that were envisaged for ConEx.

In this work, we significantly simplify ABC, improve its understanding with a comprehensive performance evaluation, and add support for traffic prioritization. The latter is designed such that transmission of prioritized traffic reduces a user's fair share to disincentivize prioritization for non-realtime or bulk traffic. Moreover, a simple extension of a static priority scheduler is proposed to avoid starvation of best effort traffic through prioritized traffic. CSFQ is a control method similar to ABC, but significantly differs in signalled information and algorithmic approach. Therefore, we compare its performance to the one of ABC and analyze why ABC achieves better fairness under challenging conditions.

The remainder of the paper is structured as follows. Section II gives an overview of congestion management approaches. In Section III we propose the new ABC design including traffic prioritization. In Section IV we use packet-based simulation to evaluate the performance of ABC, in particular fairness aspects, with emphasis on traffic prioritization. We compare the performance of CSFQ with the one of ABC in Section V. Section VI draws conclusions and gives an outlook on future work.

II. Related Work

An excellent and extensive overview of congestion management is compiled in [4]. Congestion management techniques comprise packet classification, admission control and resource reservation, caching, rate control and traffic shaping, routing and traffic engineering, packet dropping and scheduling, and many more technology-specific approaches. Moreover, multiple examples of congestion management practices applied by ISPs are described.

The whitepaper in [5] proposes that congestion management should be applied only in the presence of real congestion and discusses several detection methods for congestion that are based on time of day, concurrent user thresholds, bandwidth thresholds, and subscriber quality of experience. Application priorities and policy enforcement, e.g., prioritization, scheduling mechanisms, rate limiting, etc., are discussed as means to manage traffic when congestion is detected.

Congestion management techniques may be applied per user or per application. The latter requires deep packet inspection and expedites or suppresses traffic of certain applications. This is technically demanding, not always possible, e.g., with IPsec traffic, and widely undesired as it obviously violates network neutrality.

Rate limiting is simple and usually implemented by token-bucket based algorithms. It reduces a user's traffic rate to an upper bound regardless of the network state. Rate limiting may be applied generally or only to users that have recently been identified as heavy users, e.g., by having exceeded certain data volumes, and only for a limited time. Monthly quotas are common for many subscriber contracts. If a user exceeds his quota, his rate is throttled to an upper bound which is rather low. This is a drastic and ineffective means. As long as a heavy user has quota available, he may significantly contribute to congestion, and if his quota is consumed, he is not even able to transmit traffic in the absence of congestion.

Comcast's congestion management system [15] identifies heavy users who contribute too much traffic within a 15 minutes measurement interval. It further monitors upstream and downstream ports of cable modem termination systems (CMTSs) to detect near-congestion states. Under such conditions, the congestion management system reduces the priority of heavy user traffic to "best effort" (BE) while other traffic is served with "priority best effort" (PBO). The latter is scheduled before BE so that only heavy users suffer from lower throughput and longer delays.

Scheduling mechanisms such as WFQ, possibly approximated by Deficit Round Robin (DRR) [16], reduce a heavy user's throughput just to his fair share and only when needed. A variant of DRR allows queues to save limited deficit so that they can send faster after short idle time [17]. However, such scheduling algorithms lead to rather complex networking solutions as they require per-user queues on all potential bottleneck links, and traffic classification, which raises scalability concerns. Moreover, signalling may be needed to configure scheduling algorithms per user and to classify traffic on potential bottlenecks.

Seawall [18] is a network bandwidth allocation scheme for data centers that divides network capacity based on administrator-specific policy. It is based on congestion-controlled tunnels and provides strong performance isolation.

AQM mechanisms in routers and switches occasionally drop packets before tail drops occur. The survey in [19] gives a comprehensive overview of the large set of existing mechanisms. RED [20] was one of the first AQMs and is implemented on many routers. CoDel [21] and PIE [22] were recently discussed in the IETF AQM working group to allow temporary bursts but to avoid a standing queue and bufferbloat, in particular for links with time-varying capacity. Another AQM [23] resembles ABC in that it protects TCP traffic against non-responsive traffic, but it does not address per-user fairness. With explicit congestion notification (ECN) [24], ECN-enabled TCP senders mark packets appropriately and AQM mechanisms re-mark these packets as "congestion experienced" (CE) instead of dropping them. Thereby, ECN makes congestion visible in the network between the congested element and the receiver. Upon receipt of a CE signal, the TCP receiver signals an ECN echo (ECE) to the sender which then reduces its transmission

rate like after detection of packet loss.

Briscoe argued that per-flow fairness is not the right objective in the Internet [25] and proposed ReFeedback and congestion policing (CP) to implement per-user fairness [26]–[28]. The congestion exposure (ConEx) protocol is currently standardized by the IETF and implements the core idea of ReFeedback. ConEx leverages ECN to learn about congestion on a flow's path. A TCP sender sets for any received ECE signal a ConEx mark in the IP header of subsequent packets so that any node on a flow's path can observe its contribution to congestion. This requires modifications to TCP senders and receivers [29]. As the network cannot trust that users insert sufficient ConEx marks into packet streams, per-flow audit near the receiver should compare CE and ConEx signals to detect cheating flows and sanction them [10].

We briefly explain ConEx-based CP which is the driving application for ConEx. A congestion policer meters only ConEx-marked packets of a user and if they exceed the user's configured congestion allowance rate and burst tolerance, the policer drops some of the user's traffic. Thereby, the policer penalizes heavy users causing a lot of congestion and saves light users whose ConEx-marked traffic does not exceed their congestion allowances. The objective of this differentiated treatment is fairer bandwidth sharing among users in case of congestion. In [3] ConEx is qualitatively compared with existing congestion management approaches, and use cases are discussed which are further elaborated in [11]–[13]. There is only little insight into the performance of ConEx-based CP. Wagner [30] applied CP to a single queue. Traffic of different users is classified and separately policed before entering the queue. The policers leverage congestion information of the local queue rather than ConEx signals in data packets. This approach may be used for fair bandwidth allocation on a local queue. However, it requires per-user state so that the major advantage over scheduling is lost. ConEx Lite was developed for mobile networks in [31]. Instead of requiring users to apply ConEx, traffic is tunneled and CP is performed using congestion feedback from the tunnel. Moreover, a novel AQM presented in [32] is based on the CP principle.

ABC was inspired by ConEx-based CP but takes a different approach due to lessons learned. We have simulated ConEx-based CP and gained two insights. First, in case of congestion, policers may penalize only heavy users, but light users are also throttled by the receipt of ECE signals. Second, appropriate congestion allowances are difficult to configure. Low congestion allowances cause low utilization in case of only a few users. Large congestion allowances cannot enforce fair capacity sharing. Therefore, the performance of CP we considered benefits from leveraging information about bottleneck bandwidths and the number of current users for configuration of policers. However, this requires dynamic configuration which makes a system complex. These shortcomings are avoided in ABC.

With pre-congestion notification (PCN), meters and markers within a single domain re-mark high-priority packets if a near-congestion state is reached. This information is used at the edge of a DiffServ domain for admission control and flow termination [33]. While PCN meters traffic inside the network and drops packets from non-admitted or torn-down high-priority flows at the network edge, ABC meters traffic at the edge and drops packets inside a network. Thus, although PCN and ABC look similar at first sight, they are significantly different regarding operation and objectives.

Core-Stateless Fair Queueing (CSFQ) was the first approach to enforce fair resource sharing per flow without per-flow states in the network core [34], [35]. It is designed for a domain context. Edge nodes measure the time-dependent rates of flows and record these values as labels in the packet headers. Edge and core nodes estimate the time-dependent rate of arrived and accepted traffic $\hat{A}$ and $\hat{F}$ and estimate on that base including the known link bandwidth the fair per-flow traffic rate $\hat{\alpha}$. They drop packets with a probability of $max(0, 1 - \frac{\alpha}{p.label})$ where $p.label$ is the label of a packet p. As a consequence, packets from flows with a rate larger than the estimated fair rate $\hat{\alpha}$ are dropped so that their rate is reduced to approximately $\hat{\alpha}$.

Many improvements and variants of CSFQ have been proposed, e.g., the work in [36], [37]. The core-stateless scheduling method in [38] guarantees throughput for reserved flows. It may be applied for prioritized realtime flows in an Integrated Services (IntServ) context [39]. It is based on similar ideas as CSFQ. ABC is unaware of reservations and rather fits in a Differentiated Services context [40]. The Enhanced CSFQ with Multiple Queue Priority Scheduler provides increased rates for realtime flows by enqueuing their packets into multiple queues that are served round-robin [41]. It may create packet reordering for unequal packet sizes. The weighted variant of CSFQ [34], [35] already supports unequal flow rates by application of weights. ABC as presented in this paper provides a delay advantage with possibly adapted throughput for realtime flows, which is a different form of priority.

Rainbow Fair Queueing (RFQ) is a major variant of CSFQ [42], [43]. It assigns packets of a flow to different layers (colors) such that the "highest" color of a flow indicates its approximate rate. A simplified example: a packet rate up to 100 kb/s is marked 0, a packet rate up to 900 kb/s is marked 1, a packet rate up to 9 Mb/s is marked 2, a packet rate up to 90 Mb/s is marked 3, etc. Then the largest color of a 1 Mb/s flow is 1 and the largest color of a 10 Mb/s flow is 2. Edge and core nodes drop packets if the buffer overflows or if packets have a color larger than a certain color threshold. This color threshold is continuously updated, i.e., it is decreased if the queue size exceeds another fixed threshold, and it is increased when the link is underutilized for a while. Simulation results show that RFQ improves the performance of CSFQ with regard to fairness. Due to the integral number of layers, maintaining a high resource utilization is a problem as edge and core nodes drop all packets with a color larger than the threshold.

"Per Packet Value" (PPV) [44] can be considered as an improvement of RFQ. Packet markers at edge nodes add "per-packet values" to packets similar to RFQ. However, with

RFQ priority decreases with increasing color while with PPV priority increases with increasing per-packet value. AQMs in edge and core nodes forward only packets with the highest per-packet values in case of congestion. That means, an incoming packet replaces the packet with the lowest per-packet value in the queue if the buffer is full. Sophisticated ideas on packet marking are presented that address the utility of various flows, users, or applications. Moreover, an extension towards different delay classes is sketched. The paper [44] provides only a few performance results, but results for different delay classes were not presented.

As the simple AQM in [44] is a challenge for implementation, the authors of [45] substitute it by PVPIE which is an extension of the PIE AQM [22]. They replicate the simulation results of [44] to show that PPV works also well with PVPIE. PIE distinguishes from many other AQMs by coping with varying bandwidths. Simulation results show that PPV with PVPIE also copes with varying bandwidths. The paper points out that setting appropriate parameters for PVPIE is crucial.

ABC has been initially proposed in [14], but it is significantly simplified and extended in Section III of this paper. It differentiates from the above mentioned approaches by utilizing in the packet header an activity value which is the logarithm of the measured per-flow rate. The novel ABC's activity AQM essentially drops a packet whose activity value exceeds a certain threshold in the presence of congestion. This threshold is dynamic and depends both on the packet's activity value and the average observed activity. The first ensures that low-activity traffic enjoys lower drop precedence than high-activity traffic, giving priority to low-activity flows without complex queue management. The latter ensures that traffic is dropped only if the queue is sufficiently long, no matter whether the traffic consists of lots of low-activity flows or only a few high-activity flows. We propose two different activity meters: the normal activity meter and the fair activity meter. The normal activity meter is similar to the way traffic is marked with CSFQ as the overall rate of a flow determines the packet values. The fair activity meter is similar to the way traffic is marked with RFQ or PPV as small traffic rates from high-bitrate flows are marked with light/dark color or low activity, respectively. Furthermore, we propose support for scheduling (delay) priority for ABC and provide simulation results.

III. ABC Design

We first give an overview of ABC. Then, we explain the operation of activity meters located in edge nodes and the operation of activity-based AQMs located in edge and core nodes, pointing out the difference to the original ABC design [14]. We suggest how priority scheduling may be added. Finally, we discuss deployment aspects.

A. Overview

ABC enforces fair resource sharing within an ABC domain. The concept is illustrated in Figure 1. An ABC domain is confined by edge nodes with activity meters. They meter the activity of traffic aggregates (aka users) and mark it in any packet header. Both edge nodes and core nodes are forwarding nodes. They run activity AQMs per interface that drop traffic in case of congestion. Activity AQMs leverage the current queue length and the packets' activity to preferentially drop traffic from more active users. Therefore, ABC does not require per-flow or per-user state or signalling in the core network.

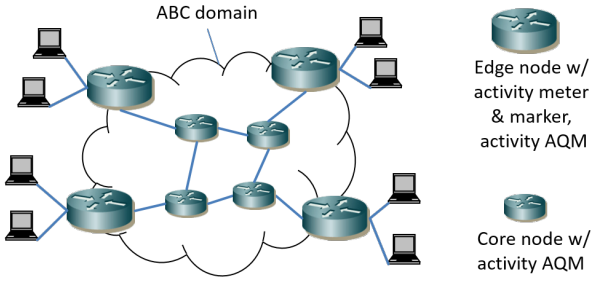


Fig. 1. Activity metering and marking is performed only at the network edge while activity AQMs may be applied on all interfaces within an ABC domain.

An activity meter is configured with a reference rate R_r. The activity meter measures the current traffic rate R_m of the monitored traffic aggregate with the arrival of each packet, computes the activity $A = \log_2(\frac{R_m}{R_r})$, and adds it to the packet header.

An activity AQM accounts for the average activity A_{avg} of accepted packets. When the activity AQM receives a packet, it reads the activity A from the packet header and decides whether to accept or drop the packet based on A, A_{avg}, and the current queue length Q.

In the following, we explain activity meter and activity AQM in more detail.

B. Activity Meter

The activity meter is configured with a reference rate R_r and measures a user's current traffic rate R_m to compute activity values. In this work, we use the TDRM-UTEMA method [46] for packet-based rate computation of R_m. The measurement starts at time t_0 with a weighted sum $S = 0$ of packet sizes and last update instant $t_{last} = t_0$. At the transmission of a new packet at time t, the weighted sum S is updated with the length L of the packet. A weighted measurement time T is derived and the measured rate R_m is calculated as fraction of the weighted sum S and the weighted time T. Thus, the following computations are performed for each packet arrival at time t:

$$S = e^{-\frac{1}{M_{AM}}\cdot(t-t_{last})}\cdot S + L \quad (1)$$

$$T = M_{AM}\cdot\left(1 - e^{-\frac{1}{M_{AM}}\cdot(t-t_0)}\right) \quad (2)$$

$$R_m = \frac{S}{T} \quad (3)$$

$$t_{last} = t \quad (4)$$

When the rates of several flows are measured, separate variables S, T, and t_{last} are needed for every flow. The advantage of this measurement approach is that it depends only on a

single parameter, the activity meter memory M_{AM}. The activity meters of all users should be configured with the same memory M_{AM}.

There are approximations for that method to avoid per-packet computation of the exponential function [46]. Finally, the activity of the packet is

$$A = \log_2\left(\frac{R_m}{R_r}\right) \tag{5}$$

and this value is recorded in the packet header.

While this activity meter can produce negative activity values if the current measured rate R_m is lower than the user's reference rate R_r, the token bucket based activity meter presented in [14] returns only activity values of at most zero. The new approach allows scaling the reference rates of all users with the same factor without changing ABC's behavior (see Section IV-E2). This makes the new ABC variant easier to configure.

C. Activity AQM

ABC forwarding nodes compute a time-dependent average activity A_{avg} from the activities A of accepted packets. The activity AQM determines based on this average A_{avg}, the activity A of a received packet, and the current queue length Q whether the received packet is accepted or dropped. In the following we explain the activity averager to calculate A_{avg}, the acceptance algorithm, and differences to the original ABC design in [14].

1) Activity Averager: The activity averager uses the UTEMA method [46] to compute the average activity A_{avg} from the activities A of accepted packets. The algorithm starts at time t_0 with a weighted sum $S = 0$, a weighted number $N = 0$, and a last update instant $t_{last} = t_0$. If a new packet is received and accepted at time t, the activity averager updates these values by

$$S = e^{-\frac{1}{M_{AA}}\cdot(t-t_{last})}\cdot S + A \tag{6}$$

$$N = e^{-\frac{1}{M_{AA}}\cdot(t-t_{last})}\cdot N + 1 \tag{7}$$

$$A_{avg} = \frac{S}{N} \tag{8}$$

$$t_{last} = t \tag{9}$$

An advantage of that method is its dependence on the single memory parameter M_{AA}. The activity averagers of all forwarding nodes should be configured with the same M_{AA}, but that value may be different from the activity meter memory M_{AM}. Moreover, M_{AA} should be smaller than M_{AM} to ensure that A_{avg} reflects current activities.

2) Acceptance Algorithm: The objective of the activity AQM is to preferentially drop packets with higher activity values A in case of congestion. Thereby "large" is relative to the average activity A_{avg}. To that end, we suggest a drop threshold which is a function of a packet's activity A and the queue length Q:

$$T_{drop}(A) = \max\left(Q_{min}, Q_{base} - \gamma\cdot(A - A_{avg})\right). \tag{10}$$

If a packet with activity A arrives in the presence of a queue length Q, the packet is dropped if the queue size is not smaller than the drop threshold $T_{drop}(A)$. The drop threshold essentially shortens the queue size for packets with an activity that is relatively large compared to most other packets observed on the link. The differentiation factor γ controls this size reduction and the parameter Q_{min} avoids packet loss in case of short queue length. The activity AQM may be adapted to time-varying links by substituting quantities and thresholds like queue sizes and Q_{min}, Q_{base}, and γ by expected queueing delay equivalents.

3) Comparison to Previous Work: The original activity AQM in [14] was significantly different: (1) A simple exponential moving average (EMA) [46] was used as activity averager. It was configured without the notion of a memory, which makes it difficult to configure. (2) The average activity A_{avg} was computed based on the activities of all received packets instead of accepted packets. (3) The acceptance algorithm modified the output of some probability-based AQM using the activity difference $(A - A_{avg})$ and the current queue length Q. The original activity AQM was rather difficult to understand, had more parameters, and essentially approximated the new method.

With the old ABC design, an excessive rate of high-activity traffic which cannot be accommodated by the link could effect that the average activity A_{avg} was close to the high activity. As a result, the high-activity traffic was dropped with too little probability, which is a particular problem when high-activity traffic is prioritized. The same effect cannot happen with the new activity averager and acceptance algorithm.

D. Extension to Traffic Classes

We extend ABC to support traffic classes with different priorities. In this work, we consider only two per-hop behaviours (PHBs, aka traffic classes): Best Effort (BE) and Expedited Forwarding (EF) [47]. However, the presented principle can be extended to more than two PHBs. It includes adaptation of the activity meter in edge nodes and adaptation of the buffer management and packet scheduling in forwarding nodes.

1) Adaptation of Activity Meters: The activity meter in edge nodes may be extended to traffic classes by multiplying the packet lengths L by a PHB-specific accounting factor a_{PHB} in Equation (1). As an example, $a_{BE} = 1$ and $a_{EF} = 3$ may be used for BE and EF. This is a means to disincentivize the use of EF for bandwidth-intense services that do not require low delay. This feature is important as EF traffic should constitute only a minor fraction of the overall traffic to ensure short queueing delay for prioritized traffic and avoid starvation of low-priority traffic.

2) Adaptation of Buffer Management and Packet Scheduling: We suggest that traffic waiting in a forwarding node is stored in PHB-specific queues. This requires adaptation of the acceptance algorithm to multiple queues and introduction of a special priority scheduling mechanism.

We integrate multiple queues into ABC by using the sum of all queue lengths as input to the activity AQM in Sec-

tion III-C2. In our special case, this is $Q = Q_{BE} + Q_{EF}$ where Q_{BE} and Q_{EF} are the lengths of the queues for BE and EF traffic.

We first consider simple priority scheduling for serving the queues. It leverages two first-in-first-out queues, one for high-priority traffic and one for low-priority traffic, and serves the high-priority queue strictly before the low-priority queue. In such a system, BE traffic may starve in the presence of a large rate of EF traffic, which continuously keeps the EF queue filled. To avoid that problem, we apply MEDF scheduling [48]. That means, a packet p is time-stamped with $p.timestamp$ before being enqueued in its PHB-specific queue. On dequeue, the packet with the lowest $p.timestamp + \Delta_{PHB}$ is chosen whereby Δ_{PHB} is a PHB-specific time offset. In our study, we use $\Delta_{EF} = 0$ and $\Delta_{BE} = \delta_{BE} \cdot \frac{Q_{max} \cdot MTU}{C_b}$ with $\delta_{BE} = 2$.

E. *Fair* Activity Meter

The normal activity meter defined in Section III-B determines the activity of an entire traffic aggregate based on its rate. As a result, the traffic of an aggregate with a larger rate has a larger activity than a traffic aggregate with lower rate, and therefore, all its traffic receives worse treatment by an activity AQM. To avoid that, we propose a fair activity meter as an alternative to the normal activity meter. It multiplies the measured rate R_m with a fractional random number $x \in (0,1)$ before calculating Equation (5). This leads to heterogeneous activity values within an aggregate and ensures that equal-rate subsets of differently large traffic aggregates exhibit similar activity values. As a consequence, equal-rate subsets of these traffic aggregates receive equal treatment by activity AQMs. The fair activity meter resembles the marking in RFQ [42], [43] and PPV [44].

We show in Section IV-H that the fair activity meter leads to fairer sharing for non-responsive traffic than the normal activity meter. However, it also degrades the fairness for responsive traffic and limits the protection of responsive traffic against non-responsive traffic. Therefore, we consider it less attractive and use the normal activity meter as default in our study.

F. Deployment Aspects

ABC requires the following network-wide parameters: memory M_{AM} for activity meters, memory M_{AA} for activity averagers, and Q_{min}, Q_{base}, and γ for configuration of activity AQMs. The priority scheduler is configured with PHB-specific offsets Δ_{PHB}. Moreover, every user needs to be configured with a reference rate R_r. These values may be user-specific for service differentiation.

The activity information may be encoded, e.g., into IPv6 extension headers or in additional headers below IP. The latter seems doable in software-defined networks, e.g., on the base of P4 [49]. As fraudulent users may set too low activity values, we propose the use of ABC only for trusted networks where the operater has control over access devices performing activity metering and marking and the transport devices.

We discuss the placement of activity meters within an ABC domain. Upstream raffic sent by a user should be activity-metered by a single entity close to the user. In contrast, the user's downstream traffic needs to be activity-metered when entering the ABC domain. If the traffic enters the ABC domain only through a single ingress node, all the traffic of the user can be activity-metered by a single entity. If the traffic enters through different ingress nodes, all the traffic of the user may be forwarded to the user's activity meter before traversing any bottleneck links. Thus, the activity meter may be considered a network service function. As an alternative, the traffic may be activity-metered by distributed activity meters. These activity meters should be configured with a partition of the user's reference rate R_r. A similar problem is distributed policing which has been demonstrated in [50]. Further details need to be discussed in the context of specific use cases which may be similar to those of ConEx-based CP [12], [13], [51].

ABC is designed to enforce per-user fairness in a network without per-user states. However, it may also be leveraged for enforcing rate fairness among multiple users served by a single interface. WFQ seems the natural solution to that problem. However, heavy users sending more traffic obtain a larger capacity share than light users sending less traffic because WFQ collects transmission permission only in the presence of waiting traffic. In case of multiple users and a small queue size, it is obvious that only a few users can have waiting packets. Under such conditions, light users are heavily disadvantaged if they maintain only a single TCP connection [17]. With ABC, activity-metering may be applied to traffic on a per-user basis before entering the queue and acceptance by the queue may be determined by an activity AQM. Thereby, similar per-user fairness is expected for TCP users as reported in Section IV.

IV. Performance Evaluation of ABC

In this section, we investigate bandwidth sharing with ABC using stochastic discrete-event simulation. We first explain the evaluation methodology and performance metrics. Then, we evaluate ABC for increasingly unequal user load, for different traffic types, without and with prioritization. We study the impact of system parameters to provide recommendations for configuration: reference rate R_r, activity AQM parameters Q_{min}, Q_{base}, and γ, and the memories M_{AM} and M_{AA} for the activity meter in edge nodes and activity averagers in forwarding nodes. We illustrate that ABC can expedite sporadic uploads in the presence of background traffic. Finally, we compare the fairness obtained with the fair activity meter and the normal activity meter.

A. Simulation Methodology

We present the simulation framework and the used traffic sources, explain the network and experiment design, introduce performance metrics and report about simulation accuracy.

1) Simulator and Traffic Sources: We performed packet-based simulation with ns3 in version 3.26. It is a "discrete-event network simulator for Internet systems" [52]. A maxi-

mum transfer unit (MTU) of 1500 bytes is supported. Packets have a 2 bytes PPP header and a 20 bytes IP header. UDP packets carry an 8 bytes UDP header and TCP packets a 32 bytes TCP header which includes 12 bytes options. Thus, the maximum segment size (MSS) for UDP is set to 1470 bytes and the MSS for TCP is set to 1446 bytes. We use the TCP New Reno implementation provided by ns3 for simulation of TCP traffic. UDP traffic sources send Poisson traffic with maximum-size UDP packets and exponentially distributed inter-arrival times T_{IAT}. Thus, the transmission rate of a UDP source is $R_t = \frac{MSS+30 \text{ bytes}}{E[T_{IAT}]}$ with an average inter-arrival time $E[T_{IAT}]$.

In previous work [14], we considered constant bit rate (CBR) traffic instead of Poisson traffic for UDP sources. However, this leads to simulation artifacts in some scenarios without ABC due to non-realistic exactness in simulations. The rate of Poisson traffic slightly varies over time. Its coefficient of variation c^I_{var} depends on the measurement interval I and can be computed by $c^I_{var}(R^i_t) = \left(\frac{R^i_t}{L} \cdot I\right)^{-1}$. For $I = 1$ s and $R^i_t = 1$ Mb/s ($R^i_t = 10$ Mb/s) we obtain $c^I_{var}(R^i_t) = 0.11$ ($c^I_{var}(R^i_t) = 0.03$). Thus, the resulting rate variations are rather small.

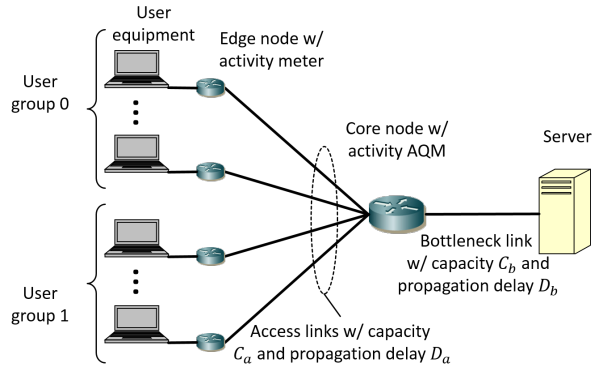


Fig. 2. Network and experiment design.

2) Network Design: We simulate the network depicted in Figure 2. Multiple users communicate with a server via an edge node, an access link, and a bottleneck link, yielding a one-sided dumbbell topology. All access links have the same propagation delay D_a and bandwidth C_a. The bottleneck link is shared by all users, has propagation delay D_b and capacity C_b. Thus, a lower bound for the round trip time (RTT) is $2 \cdot (D_a + D_b)$.

On the fast access links, we apply drop-tail queues with a buffer size of 50 packets which are hardly filled. The bottleneck link is implemented as a packet queue, i.e., every packet requires one slot in the queue independently of its size. Therefore, queue lengths and thresholds are given in packets (pkts). It has a transmission buffer of up to $Q_{max} = 24$ pkts. In case of ABC, the bottleneck link features an activity AQM with parameters $Q_{min} = 12$ pkts, $Q_{base} = 20$ pkts, $\gamma = 16$ pkts, and a memory of $M_{AA} = 0.3$ s for its activity averager.

3) Experiment Design: Figure 2 shows that users are divided into a user group 0 (UG_0) and a user group 1 (UG_1) in our experiments. If the experiments require a distinction between heavy and light users, the heavy users (HUs) are grouped in UG_0 and the light users (LUs) in UG_1. The user groups UG_i have u_i users whose traffic is monitored by separate activity meters which are all configured with the same reference rate R^i_r. All users of a user group communicate in the same way with the server. That means, in case of TCP communication, each user in UG_i has f_i TCP flows. In case of non-responsive traffic, a user has only a single UDP flow sending Poisson traffic at a transmission rate of R^i_t.

ABC's intention is to share bandwidth in case of congestion proportionally to the users' reference rates. We study to which degree this objective can be achieved with different traffic types. To consider challenging conditions, we model heavy users competing with light users.

To study non-responsive traffic, we simulate a single user per user group UG_i with a single flow sending Poisson traffic at rate R^i_t. If two non-responsive users compete for the bandwidth of a link without ABC, the bandwidth is shared proportionally to the transmission rates. This yields a configured unfairness of $\frac{R^0_t}{R^1_t}$.

To study responsive traffic, we simulate multiple users with various numbers of saturated TCP flows, i.e., flows that have always data to send. More specifically, we simulate one heavy and one or multiple light TCP users with f_0 and f_1 TCP flows ($f_0 \geq f_1$), respectively. TCP flows share the capacity of a bottleneck link about equally. Thus, if heavy and light users compete for the bandwidth of a link without ABC, the bandwidth is shared proportionally to their numbers of flows. This yields a configured unfairness of $\frac{f_0}{f_1}$.

We also consider the coexistence of heavy users with non-responsive traffic and light users with non-responsive traffic.

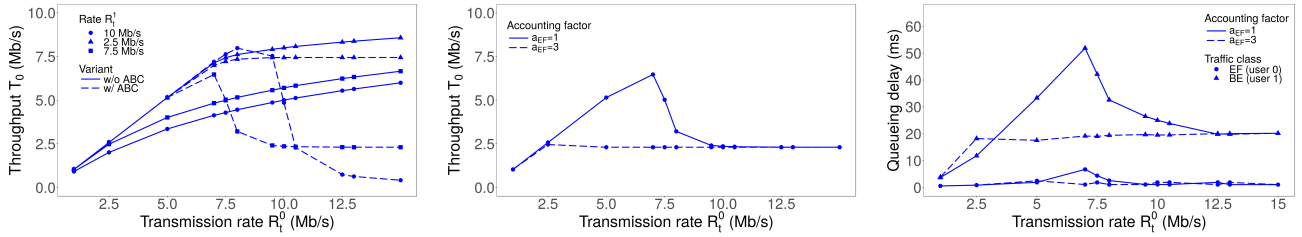
When we investigate delay prioritization, we utilize an accounting factor of $a_{BE} = 1$ for BE traffic and $a_{EF} = 3$ for EF traffic, and use $\delta_{BE} = 2$ to calculate the BE-specific time offset Δ_{BE} for the scheduler (see Section III-D2).

We perform multiple experiments that differ only in a few parameters. Table I compiles default values that are applied in all experiments if not stated differently.

TABLE I
DEFAULT PARAMETERS.

link bandwidths C_a, C_b	100 Mb/s, 10 Mb/s
link delays D_a, D_b	0.1 ms, 5 and 50 ms
buffer size Q_{max} on bottleneck link	24 pkts
activity AQM parameters Q_{min}, Q_{base}, γ	12, 20, 16 pkts
activity meter and averager memory M_{AM}, M_{AA}	3 s, 0.3 s
users u_0/u_1 in $UG_{0/1}$	1/10
flows f_0/f_1 per user in $UG_{0/1}$	10/1
reference rates R^i_r	10 kb/s
accounting factors a_{BE}, a_{EF}	1, 3
δ_{BE} for configuration of Δ_{BE}	1.5

4) Performance Metrics: The widely used fairness index of Jain [53] is at most 1 and does not reveal the advantaged



(a) Throughput T_0 of user 0 – no prioritization.

(b) Throughput T_0 of user 0 – traffic of user 0 is prioritized and transmission rate of user 1 is $R_t^1 = 7.5$ Mb/s.

(c) Avg. queuing delay for BE and EF traffic – traffic of user 0 is prioritized and transmission rate of user 1 is $R_t^1 = 7.5$ Mb/s.

Fig. 3. User 0 and user 1 send Poisson traffic and their activity meters are configured with equal reference rates $R_r^i = 10kb/s$.

user group. Therefore, we characterize the fairness of the bandwidth sharing result by the throughput ratio $T_R = \frac{\overline{T_0}}{\overline{T_1}}$ where $\overline{T_i}$ is the average throughput per user in UG_i. The bandwidth is shared fairly among users in both groups if $T_R = 1$ holds. In case of $T_R > 1$, users in UG_0 receive more throughput than users in UG_1, and in case of $T_R < 1$, users in UG_1 are advantaged. Thus, maximizing per-user fairness means bringing T_R close to 1.

For some experiments, we also report the average queue length or the average queueing delay, and the long-term utilization of the bottleneck link.

5) Simulation Accuracy: If not stated differently, flows start uniformly distributed within the first 15 s of a simulation run. The first 100 s of a simulation run are discarded and results are logged over the consecutive 100 s. We report mean values over 10 runs. We omit confidence intervals for the sake of better readability.

B. Performance with Non-Responsive Traffic

In a first experiment series we show how two competing users sending Poisson traffic share the bandwidth of a bottleneck link without and with ABC. We set the transmission rate of user 1 to $R_t^1 \in \{2.5, 7.5, 10\}$ Mb/s and vary user 0's transmission rate R_t^0 to study his resulting throughput T_0. We perform experiments without and with prioritization. Results are shown for a one-way delay of $D_b = 5$ ms only, because that parameter has no impact without reactive sources.

1) Without Prioritization: We first consider the case without prioritization. Figure 3(a) shows the throughput T_0 of user 0 without and with ABC. Without ABC, the throughput T_0 of user 0 increases with increasing transmission rate R_t^0. User 0 even benefits from overloading the link. As soon as the network is overloaded, the queue is fully filled and the utilization is 100% (without figures).

For ABC we set the reference rates of both users to $R_r^i = 10$ kb/s. As long as the link is not overloaded, the throughput values for both users are the same without and with ABC. Therefore, the throughput of user 0 increases with increasing transmission rate R_t^0 up to a value of $T_0 = 7.5$ Mb/s in case of ABC and $R_t^1 = 2.5$ Mb/s. It remains constant for larger transmission rates R_t^0 so that user 1 gets a throughput of $T_1 = 2.5$ Mb/s.

We explain the observed effect. The activities of the traffic sent by both users are depend on their transmission rates. The traffic of the user with the higher transmission rate obtains lower activity. In case of congestion, a queue builds up. The activity AQM drops traffic with higher activity values at lower queue length while its still accepts the traffic with lower activity values. Thus, the activity AQM preferentially drops traffic from the more active user, i.e., user 0, in case of congestion.

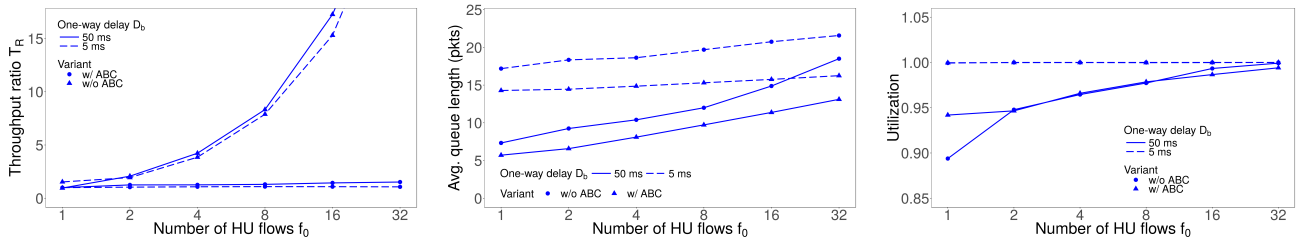
For $R_t^1 = 7.5$ Mb/s, the throughput T_0 of user 0 increases up to a transmission rate $R_t^0 = 7.5$ Mb/s and reaches a value shortly below $T_0 = 7.5$ Mb/s although the bottleneck link is already saturated at $R_t^0 = 2.5$ Mb/s. For a transmission rate of $R_t^0 < 7.5$ Mb/s, packets of user 0 have lower activity than those of user 1. Therefore, packets of user 1 are preferentially dropped so that the throughput T_0 of user 0 increases. For $R_t^0 > 7.5$ Mb/s the throughput of user 0 falls down to $T_0 = 2.5$ Mb/s so that user 1 gets a throughput of $T_1 = 7.5$ Mb/s. This is because then packets of user 0 have larger activity and are preferentially dropped.

For $R_t^1 = 10$ Mb/s we see the same phenomenon, which means that user 0 obtains the full link capacity if user 0 exceeds his transmission rate beyond 10 Mb/s.

With ABC, packets are already dropped in case of congestion although they could be accommodated in the buffer. The activity AQM mostly enforces queue lengths lower than Q_{base} (without figures) so that the average queue length is significantly lower than without ABC. However, 100% bottleneck utilization is still achieved.

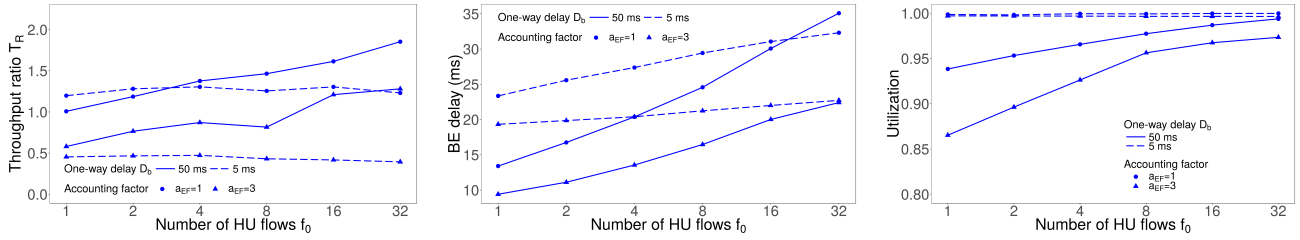
Thus, in case of equal reference rates, traffic of users with the lower transmission rates enjoy priority over traffic of users with larger transmission rates. To maximize throughput in case of congestion, users should reduce their activity. Hence, ABC provides an environment that incentivizes users to apply congestion control.

2) With Prioritization: We now perform the same experiment but prioritize the traffic of user 0 over the traffic of user 1. That means, traffic of user 0 is sent as EF, accommodated in a separate queue, and served with the priority scheme proposed



(a) Throughput ratio. (b) Average queue length. (c) Utilization of the bottleneck link.

Fig. 4. Both user 0 and user 1 send TCP traffic without prioritization. The heavy user has a various number of f_0 flows and the light user has $f_1 = 1$ flow.



(a) Throughput ratio. (b) Avg. queuing delay for BE traffic. (c) Utilization of the bottleneck link.

Fig. 5. Both user 0 and user 1 send TCP traffic, traffic of user 0 is prioritized. The heavy user has a various number of f_0 flows and the light user has $f_1 = 1$ flow.

in Section III-D2. We apply two different accounting factors $a_{EF} \in \{1,3\}$ while keeping $a_{BE} = 1$. We set the transmission rate of user 1 to to $R_t^1 = 7.5$ Mb/s. Figure 3(b) shows that the throughput T_0 of user 0 is almost identical as without prioritization for $a_{EF} = 1$. For $a_{EF} = 3$, T_0 is significantly lower than for $a_{EF} = 1$ because higher accounting factors lead to higher activity values for user 0's traffic. Thus, when applying a large accounting factor a_{EF}, a user can send less EF traffic than BE traffic in case of congestion. This disincentivizes users to send traffic without realtime requirements with EF.

Figure 3(c) compiles the average queueing delays for the traffic of both users. EF traffic of user 0 is faster forwarded on the bottleneck link than BE traffic. Therefore, the average queueing delay of EF traffic is about zero under most conditions. Only for a large EF traffic rate of $R_t^0 = 7$ Mb/s and an accounting factor of $a_{EF} = 1$, the average queueing delay for EF traffic is larger but does not exceed 5 ms. This is due to lots of accepted EF traffic under these conditions. With an accounting factor of $a_{EF=3}$, less EF traffic is accepted which then enjoys shorter queueing delay. BE traffic of user 1 waits significantly longer. The queuing delay for BE is even larger for accounting factor $a_{EF} = 1$ than for $a_{EF} = 3$ because more EF traffic is transmitted with $a_{EF} = 1$. The average queueing delay can become very large like 50 ms for $a_{EF} = 1$ although the buffer size of $Q_{max} = 24$ packets corresponds to only 31.6 ms delay. For $a_{EF} = 3$, the average queueing delay for BE traffic is limited to about 20 ms.

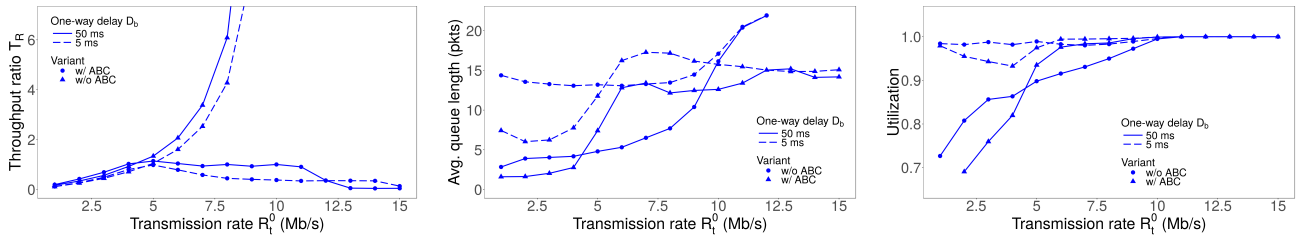
C. *Performance with Responsive Traffic*

We now consider transmission of TCP traffic. User 0 is a heavy user and opens a large number f_0 of TCP connections while user 1 is a light user and holds only $f_1 = 1$ TCP connection. Again, we investigate this scenario without and with ABC, and without and with prioritization of the heavy user.

1) Without Prioritization: Figure 4(a) shows the throughput ratio of the heavy user compared to the light user. Without ABC, the throughput ratio $T_R = \frac{T_0}{T_1}$ can be approximated by $\frac{f_0}{f_1}$. This well matches our simulation results for short and long one-way delay $D_b \in \{5, 50\}$ ms. In contrast with ABC, the throughput ratio of heavy and light users remains between 1 and 1.1. It is slightly lower for short one-way delay than for long one-way delay.

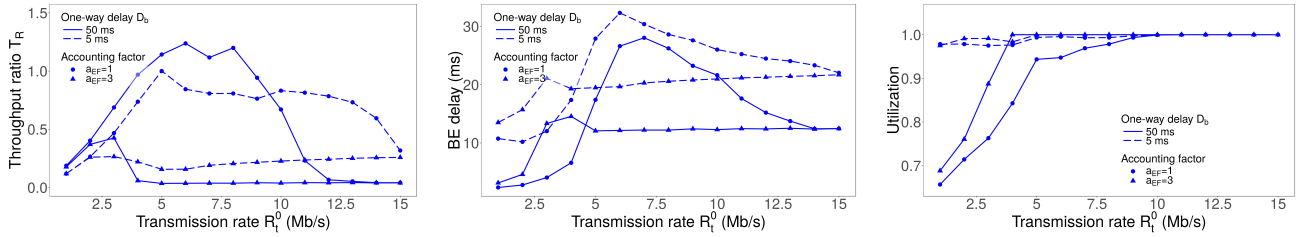
Figure 4(b) reports the average queue length. It is shorter for short one-way delay of $D_b = 5$ ms than for long one-way delay of $D_b = 50$ ms and increases with the increasing number of flows f_0. ABC leads to slightly shorter average queue length than without ABC because the activity AQM drops packets already before the queue is fully filled.

Figure 4(c) demonstrates that utilization is 100% for short one-way delay of $D_b = 5$ ms. For long one-way delay of $D_b = 50$ ms, it is lower and increases with an increasing number of flows from 92% to almost 100%. Traffic sharing with ABC leads to similar utilization as without ABC. Although ABC causes shorter average queue lengths than without ABC, it allows for almost identical link utilization except for $f_0 = 1$ and $D_b = 50$.



(a) Throughput ratio. (b) Average queue length. (c) Utilization of the bottleneck link.

Fig. 6. User 0 sends Poisson traffic with transmission rate R_t^0 and user 1 has $f_1 = 1$ TCP flow. No prioritization.



(a) Throughput ratio. (b) Avg. queuing delay for BE traffic. (c) Utilization of the bottleneck link.

Fig. 7. User 0 sends Poisson traffic with transmission rate R_t^0 and user 1 has $f_1 = 1$ TCP flow. ABC is applied and traffic of user 0 is prioritized.

2) With Prioritization: We perform the same experiments as above while prioritizing the traffic of the heavy user.

For prioritization without ABC we report results without figures. For a one-way delay of $D_b = 50$ ms, the throughput ratio increases in a very similar way as without prioritization. This is because the round-trip times for both users are in the same order of magnitude as they are dominated by the one-way delay since queuing delays for $D_b = 50$ ms are rather low. However, for $D_b = 5$ ms, observed throughput ratios start from 200. This is because the TCP connections of the heavy user experience very short round-trip times due to short one-way delay and negligible queueing delay. In contrast, the TCP connection of the heavy user experiences short one-way delay but long queueing delay, i.e., long round-trip times, which limits its throughput to low values.

Figure 5(a) shows the throughput ratio for ABC and prioritization of heavy user traffic. We consider one-way delays of $D_b \in \{5, 50\}$ ms and accounting factors $a_{EF} \in \{1, 3\}$. With accounting factor $a_{EF} = 1$, the throughput ratio remains around 1.2 for $D_b = 5$ ms and increases for an increasing number of flows f_0 of the heavy user from 1.0 to 1.7 for $D_b = 50$ ms. With $a_{EF} = 3$, the figure reports clearly lower throughput ratios because EF traffic is tagged with larger activity values at lower transmission rates. The throughput ratio remains around 0.5 for $D_b = 5$ ms and increases from 0.6 to 1.1 for $D_b = 50$ ms. Thus, prioritizing a user's traffic increases his throughput ratio, but applying an appropriate accounting factor a_{EF} counteracts that phenomenon.

In all cases and under all conditions, EF traffic experiences an average queueing delay close to zero. Figure 5(b) reports the average queuing delay for BE traffic. With $a_{EF} = 1$, it increases from 24 ms to 31 ms for $D_b = 5$ ms, and from 14 ms to 35 ms for $D_b = 50$ ms. With $a_{EF} = 3$, less EF traffic is sent, which reduces the queueing delay for BE traffic to values between 19 ms and 21 ms and between 9 ms and 21 ms, respectively.

Figure 5(c) reports the utilization of the bottleneck link for ABC with traffic prioritization, again for $D_b \in \{5, 50\}$ ms and for $a_{EF} \in \{1, 3\}$. For $D_b = 5$ ms, 100% utilization is achieved for $a_{EF} = 1$ and slightly less for $a_{EF} = 3$. For $D_b = 50$ ms, utilization values increase from 94% to 100% with increasing number of flows and are about equally high as without prioritization (see Figure 4(c)). We observe the same effect without ABC and with prioritization (without figure). A larger accounting factor of $a_{EF} = 3$ reduces the utilization compared to $a_{EF} = 1$.

D. Performance with Non-Responsive and Responsive Traffic

We investigate the coexistence of a non-responsive user and a responsive user. The non-responsive heavy user sends Poisson traffic with a transmission rate of R_t^0 and the light responsive user transmits a single TCP flow. We study the throughput ratio T_R without and with ABC and without and with prioritization of the heavy user.

1) Without Prioritization: Figure 6(a) shows the throughput ratio of both users depending on the transmission rate of the heavy user. Without ABC, the heavy user obtains a major capacity share of the bottleneck link for transmission rates R_t^0 larger than 5 Mb/s. Figure 6(b) illustrates that as soon as

a transmission rate of $R_t^0 = 10$ Mb/s is reached, the queue is mostly fully occupied. According to Figure 6(c), the link utilization is close to 100% for a one-way delay of 5 ms for any transmission rate R_t^0. In contrast, for a long one-way delay of $D_b = 50$ ms, the utilization is much lower and increases from 75% to 100% with increasing transmission rate R_t^0.

With ABC, the throughput ratio between the heavy and light user in Figure 6(a) rises up to a value of 1.2 and remains at this value for a broad range of transmission rates R_t^0 before it falls to almost 0. This can be observed for one-way delays of both $D_b = 5$ ms and $D_b = 50$ ms to a different extent. The heavy user's traffic is preferentially dropped in case of congestion because of larger activity values. The rate of the light TCP user cannot increase significantly beyond 10 Mb/s. Thus, when the heavy user's rate R_t^0 is larger than 10 Mb/s, the light user's traffic is always prioritized over the heavy user's traffic. However, the heavy user is then not completely starved because some of its traffic can be transmitted when the single flow of the light user cannot keep the queue length so large that all traffic of the heavy user is dropped. This critical queue length increases with increasing transmission rate R_t^0.

In early experiments, we used plain priority scheduling for the differentiation of EF and BE traffic. Then BE traffic was starved if the EF traffic rate R_t^0 exceeded 10 Mb/s. This happened because the EF queue was mostly filled so that BE traffic was rarely scheduled. The modified priority scheduling approach presented in Section III-D2 fixed that problem.

With ABC, the average queue length in Figure 6(b) stays significantly shorter for $R_t^0 > 10$ Mb/s than without ABC because traffic from user 0 is already dropped in the presence of a short queue due to high activity values. Figure 6(c) shows that the utilization with ABC is lower than the utilization without ABC, but for large R_t^0 the utilization is larger than the utilization without ABC. Overall, it is similar.

Most notably in this experiment is that with ABC, non-responsive users cannot dominate responsive users while they do without ABC.

2) With Prioritization: We perform the same experiments as above with prioritization of the heavy user's traffic. We first consider the experiment without ABC. In this case, the throughput ratio increases with prioritization (no figure) even faster than without prioritization (see Figure 6(a)), i.e., the TCP traffic of the light user is even more suppressed by the heavy user's traffic as it suffers from increased round-trip times due to increased queuing delay. We now consider transmission with ABC and prioritization in Figure 7(a). With an accounting factor of $a_{EF} = 1$, we obtain similar results as with ABC but without prioritization in Figure 6(a). With an accounting factor of $a_{BE} = 3$ we observe significantly lower throughput ratios because the larger accounting factor increases the activity values of the heavy user's traffic.

As EF traffic is prioritized, the heavy user's traffic hardly faces any queueing delay. Figure 7(b) reports that the average queuing delay for BE traffic rises up to a value of around 30 ms for accounting factor $a_{EF} = 1$. It falls again for larger transmission rates because then only little EF traffic is admitted to the system due to its large activity values. For accounting factor $a_{EF} = 3$, the activity AQM accepts less EF traffic. Therefore, BE traffic faces lower average queueing time.

ABC with prioritization and accounting factor $a_{EF} = 1$ does not decrease the link utilization compared to capacity sharing without ABC and prioritization (without figure). Figure 7(c) shows that $a_{EF} = 3$ leads to increased utilization in case of small transmission rate R_t^0. Moreover, utilization with ABC is almost equal without and with prioritization (cf. Figure 6(c)).

E. Impact of Reference Rate R_r

Although ABC's control depends on relative reference rates, we prove that scaling them for all users with the same value yields identical behavior of the activity AQM, which also leads to identical sharing results. Furthermore, we show that unequal user-specific reference rates may be used to scale a user's throughput relative to the throughput of other users.

1) Scalability of Reference Rates: Activities are computed as $A = \log_2(\frac{R_m}{R_r})$ with measured rates R_m. We scale the reference rates of all users by a scalar s to $R_r^* = s \cdot R_r$, which yields modified activities

$$A^* = \log_2\left(\frac{R_m}{s \cdot R_r}\right) = \log_2\left(\frac{R_m}{R_r}\right) - \log_2(s) = A - \log_2(s). \tag{11}$$

The activity AQM calculates a moving average A_{avg} over the activities of previously accepted packets. Due Equation (11), the moving average over the scaled activities is

$$A_{avg}^* = A_{avg} - \log_2(s). \tag{12}$$

The drop decision of the activity AQM for a packet depends on the difference of the packet's activity A and the averaged activity A_{avg}, i.e., on $A - A_{avg}$. We obtain for that difference in the scaled system

$$A^* - A_{avg}^* = (A - \log_2(s)) - (A_{avg} - \log_2(s)) = A - A_{avg}. \tag{13}$$

Therefore, activity AQMs perform the same drop decisions when the reference rates of all users are scaled with the same value. Simulations with different scaling factors confirmed this finding. Minor deviations occurred due to numerical inaccuracies.

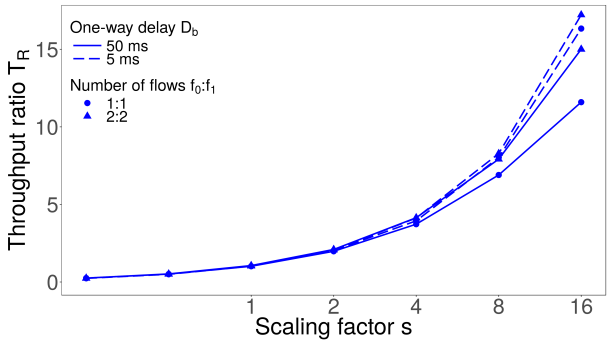
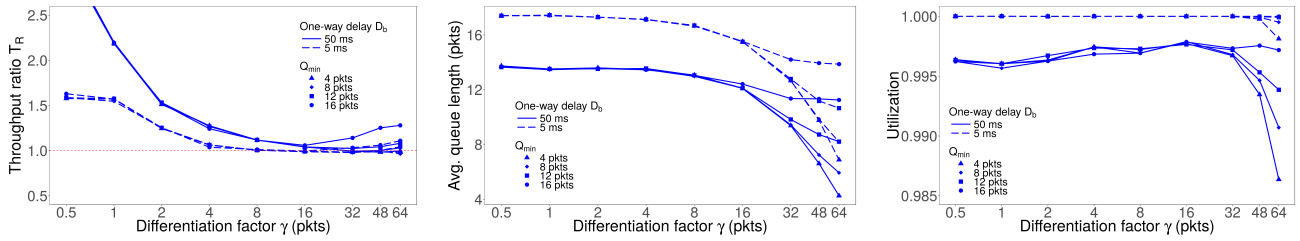
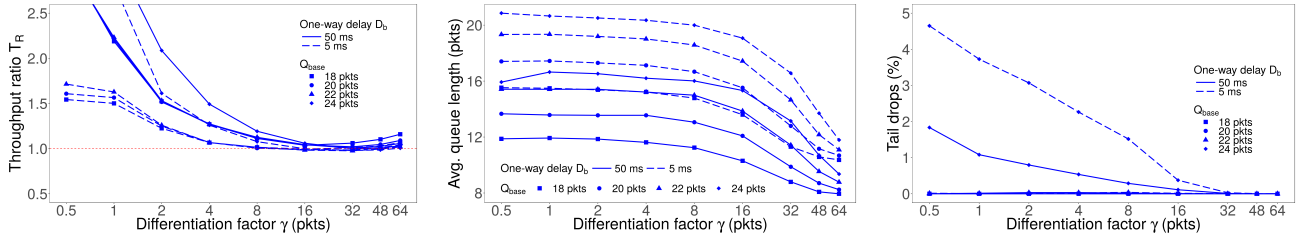


Fig. 8. Impact of up-scaled reference rates on throughput ratio.



(a) Throughput ratio. (b) Average queue length. (c) Utilization of the bottleneck link.

Fig. 9. A single heavy user sends $f_0 = 10$ TCP flows and $u_1 = 10$ light users send $f_1 = 1$ TCP flow. Impact of activity AQM parameters γ and Q_{min} for different one-way delays D_b.



(a) Throughput ratio. (b) Average queue length. (c) Percentage of tail drops.

Fig. 10. A single heavy user sends $f_0 = 10$ TCP flows and $u_1 = 10$ light users send $f_1 = 1$ TCP flow. Impact of activity AQM parameters γ and Q_{base} for different one-way delays D_b.

2) Scalability of User Shares: We study the extent to which a user's capacity share can be scaled by adapting his reference rate. We consider two users sending TCP traffic. We scale the reference rate of user 0 with a factor s, i.e., $R_r^0 = s \cdot R_r^1$, and investigate its impact on the throughput ratio.

Figure 8 shows the throughput ratio for one and two flows per user. For a short one-way delay of $D_b = 5$ ms, user 0 is able to take full advantage of his upscaled reference rate R_r^0 as the throughput ratio T_R increases linearly with the scaling factor s. We observe that also for two flows per user. In contrast, the throughput ratio is only $T_R = 12$ for a long one-way delay of $D_b = 50$ ms, a scaling factor of $s = 16$, and one flow per user. This is because a single TCP flow cannot recover fast enough from packet loss in the presence of a long round-trip time. However, for two flows per user, a throughput ratio of $T_R = 15$ is achieved, which approximates the scaling factor $s = 16$. Thus, scaling of user shares works, but with TCP traffic a sufficiently large number of flows may be prerequisite.

F. Impact of Activity AQM Parameters

The activity AQM preferentially drops traffic from heavy users in case of congestion. We evaluate the impact of its parameters γ, Q_{min}, and Q_{base} on throughput ratio, queue length, link utilization, and tail drops.

1) Impact of γ and Q_{min}: We consider a single heavy user with $f_0 = 10$ TCP flows and $u_1 = 10$ light users with $f_1 = 1$ TCP flow each. Figure 9(a) shows that small differentiation factors of $\gamma = 1$ packet already provide good fairness in terms of throughput ratio of about 2. However, larger values of γ improve the fairness towards throughput ratios between 1.0 and 1.1. Increasing γ even further, slightly increases the throughput ratio again. Short one-way delay of $D_b = 5$ ms leads to smaller throughput ratios than large one-way delay of $D_b = 50$ ms because shorter one-way delay allows TCP to faster adapt its transmission rate. The minimum threshold Q_{min} has only a minor impact on the throughput ratio and its effect is only visible for small Q_{min} and very large γ.

The differentiation factor γ essentially influences the queue limit at which packets with relatively large activity values are still accepted. Therefore, it also impacts the average queue length. Figure 9(b) shows that the average queue length is large for small differentiation factors γ while it is small for large differentiation factors γ. Short one-way delay $D_b = 5$ ms leads to larger average queue sizes than long one-way delay $D_b = 50$ ms. Again, the effect of the minimum threshold Q_{min} is only visible for small Q_{min} and very large γ.

The resulting utilization on the bottleneck link is reported in Figure 9(c). Short one-way delay of $D_b = 5$ ms leads to 100% utilization for most parameter settings while long one-way delay of $D_b = 50$ ms leads to utilization values around 99.6% which are still very high; without ABC, we observed 99.7% utilization for this setting. Very large differentiation factors γ may slightly reduce utilization, but this effect is mitigated by larger minimum thresholds Q_{min}.

Based on this investigation, we choose $\gamma = 16$ packets and $Q_{min} = 12$ packets because these values provide good fairness and avoid reduced utilization.

2) Impact of Q_{base}: We now investigate the impact of Q_{base}. Figure 10(a) shows that it has hardly any impact on the relative throughput. Only for a very large value of $Q_{base} = 24$ pkts the relative throughput is increased. Figure 10(b) illustrates that increasing Q_{base} clearly increases the average queue size as it controls the threshold from which on packets are dropped. Figure 10(c) compiles the percentage of packet drops due to buffer overflow. It is zero for almost all values of Q_{base} and γ but for $Q_{base} = 24$ pkts it is significant. As tail drops can affect packets from both heavy and light users, they reduce ABC's drop differentiation between them, which leads to increased throughput ratios. It is remarkable that ABC avoids tail drops in the presence of TCP traffic even for large values of Q_{base} as long as there is still some headroom to the queue size.

G. Impact of Memories

ABC requires a memory M_{AM} for activity meters in edge nodes and a memory M_{AA} for activity averagers in forwarding nodes. We study the impact of these memories mathematically and by two experiment series.

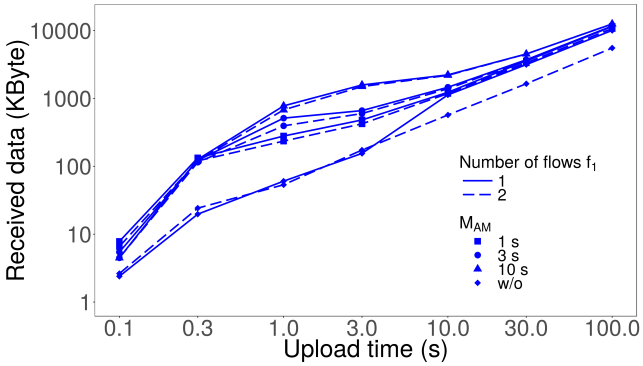
1) Activity Advantage for Starting Users: When users start sending, their initial measured rate starts with $R_m = 0$ and slowly ramps up with the amount of transmitted traffic. Therefore, a starting user enjoys lower activity values than users transmitting at similar rate, which leads to an initial activity advantage. In the following, we point our the causes of this effect.

We consider a user who continuously transmits at a transmission speed of $R_t = \frac{C_b}{2}$ with $C_b = 10$ Mb/s. According to Equation (3), the weighted sum S of his activity meter must be $S = R_m \cdot T = R_t \cdot M_{AM}$ if the measured rate R_m well approximates the transmission rate R_t and if the measurement process runs long enough so that the weighted measurement time T has converged to M_{AM}. For an activity meter memory of $M_{AM} = 3$ s, the weighted sum is 1.875 MByte.

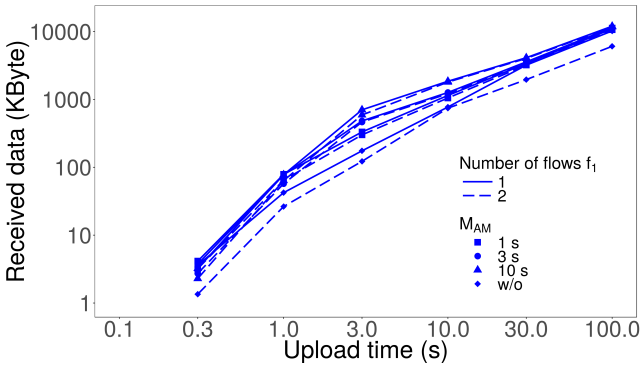
We now consider a starting user. His weighted sum is initially $S = 0$ and increases only with metered traffic. In particular, at least 1.875 MByte must be transmitted before the starting user can obtain similar activity values as the continuously transmitting user. The starting user can even send arbitrarily fast for that purpose without risking high activity values, which constitutes an initial activity advantage. That advantage scales with M_{AM}. The advantage can be compared to the tolerance of a token bucket based rate limiter which allows transmission of initial burst, which is controlled by the bucket's size.

2) Reduction of Upload Time for Starting Users: We study the practical consequences of the previously observed initial activity advantage in the following experiment. We investigate the impact of ABC on the upload speed of sporadic uploads as they occur with interactive applications. Users send small amounts of data and are then inactive until the next upload. Acceleration of such uploads can improve the user's quality of experience.

We consider an upload user who starts transmission of a large file after long idle time and in the presence of background traffic. The background traffic is generated by $u_1 = 10$ other users with $f_1 \in \{1,2\}$ saturated TCP flows each. User 0 starts uploading only 100 s after all background users have started their TCP connections to ensure that their activity meters already yield typical activities. We measure the amount of user 0's traffic that is correctly received by TCP over time.



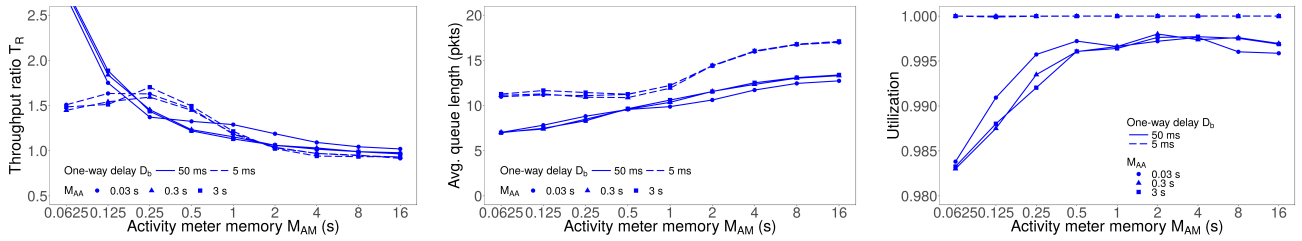
(a) One-way delay $D_b = 5$ ms.



(b) One-way delay $D_b = 50$ ms.

Fig. 11. Traffic received from an upload user depending on time. Background traffic is generated by $u_1 = 10$ users with $f_1 \in \{1,2\}$ TCP connections. All users are configured with equal reference rates.

Figures 11(a) and 11(b) compile the amount of data received from the upload user over time for various background traffic ($f_1 \in \{1,2\}$), without and with ABC, and for different one-way delay $D_b \in \{5,50\}$ ms. Note the logarithmic scale of the x-axis and the y-axis in the figures. We first discuss results in Figure 11(a) for $D_b = 5$ ms one-way delay. With ABC, significantly more traffic is uploaded after 1 s. The amount of traffic received after 1 s increases with the activity meter memory M_{AM}. The difference in received traffic is due to the initial activity advantage of the upload user. The relative difference in received traffic among different activity meter memories M_{AM} decreases over time. Also the relative difference of received traffic without and with ABC vanishes if every background user has only a single ($f_1 = 1$) TCP flow. If background users have $f_1 = 2$ TCP flows, upload with ABC remains faster than without ABC because ABC enforces fair bandwidth sharing among all users, and this advantage remains over time. Thus, ABC enforces fair capacity sharing, which is an advantage for light users against heavy users, and it favors



(a) Throughput ratio. (b) Average queue length. (c) Utilization of the bottleneck link.

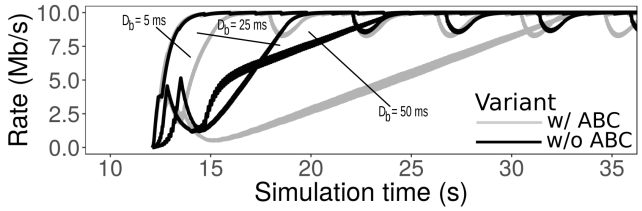
Fig. 12. A single heavy user sends $f_0 = 10$ TCP flows and $u_1 = 10$ light users send $f_1 = 1$ TCP flow. Impact of activity meter memory M_{AM} and activity averager memory M_{AA} for different one-way delays D_b.

users that have been silent for some time as they enjoy an activity advantage. Large uploads mainly benefit from the first phenomenon, small uploads also benefit from the latter.

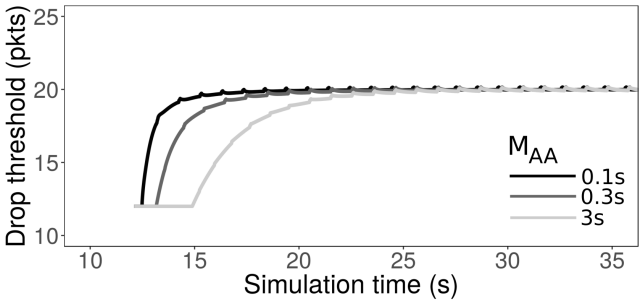
Figure 11(b) shows the upload time for one-way delays of $D_b = 50$ ms. The difference in upload time without and with ABC is smaller than for one-way delays of $D_b = 50$ ms. However,in the presence of background users with $f_1 = 2$ TCP connections, uploads with ABC are still faster than without ABC because ABC enforces fair capacity sharing.

3) Impact on Fairness: We now study the impact of activity meter memory M_{AM} and activity averager memory M_{AA} on fairness. To that end, we consider one heavy user with $f_0 = 10$ TCP flows and $u_1 = 10$ light users with $f_1 = 1$ flow and test one-way delays of $D_b \in \{5, 50\}$ ms. Their throughput ratio is presented in Figure 12(a). The fairness between the heavy user and the light users improves with increasing activity meter memory M_{AM} and good values are obtained for $M_{AM} = 2$ s and larger. Values of $M_{AM} = 0.25$ s and smaller can significantly degrade the fairness. Figure 12(b) shows that average queue lengths are larger for $D_b = 5$ ms than for $D_b = 50$ ms and that they increase with increasing activity meter memory M_{AM}. If the activity meter memory M_{AM} is large, activity values increase later compared to short M_{AM} so that activity AQMs start dropping packets also delayed, which results in increased average queue lengths. Figure 12(c) reveals that the activity meter memory M_{AM} has no detrimental influence on bottleneck utilization if it is larger than 0.5 s. In this study, we apply $M_{AM} = 3$ s to keep throughput ratios low and to ensure that activity values adapt sufficiently fast in the presence of traffic rate changes. The activity averager memory M_{AA} has hardly any impact on the performance metrics in Figures 12(a)–12(c).

4) Impact of Activity Averager Memory M_{AA}: We performed additional experiments with varying activity averager memory M_{AA} which is used by the activity AQM in forwarding nodes. It turned out that this parameter has only minor impact on system performance: minor impact on upload performance, minor impact on fairness, and minor impact on bandwidth sharing. However, it is recommendable to set it to a value smaller than the activity meter memory M_{AM} to minimize the difference between measured activities A at the edge nodes and the averaged activity A_{avg} in forwarding nodes. Given a packet transmission time of $\frac{1500 \cdot 8\text{bit}}{10^7 \text{bit/s}} = 1.2$ ms, an activity averager memory in the order of $M_{AA} = 300$ ms seems enough so that the average activity is computed over the activity of sufficiently many packets.



(a) Impact of one-way delay D_b without and with ABC.

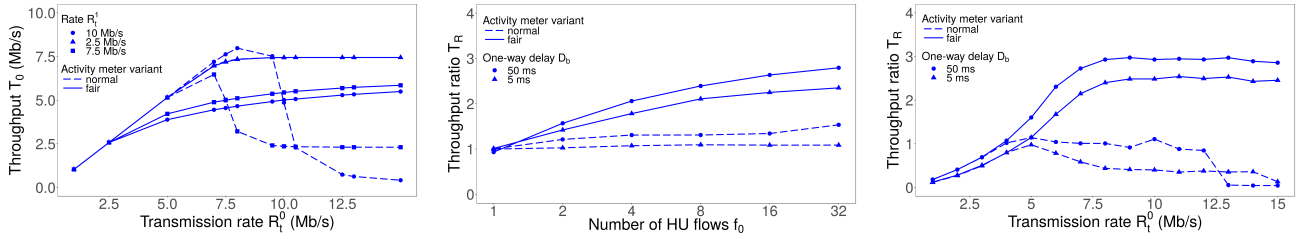


(b) Impact of activity averager memory M_{AA} on ABC's drop threshold; the one-way delay is $D_b = 5$ ms.

Fig. 13. Transmission of a single TCP flow.

5) Impact on the Initial Rate Increase of a Single TCP Flow: The activity AQM limits the queue length depending on various parameters. Therefore, a single TCP flow may not be able to fully utilize the available buffer size. This may be a problem for TCP flows in the presence of large bandwidth-delay products (BDP) and an underdimensioned buffer as its rate may take long to fully utilize the bottleneck bandwidth.

Figure 13(a) visualizes the time-dependent transmission rate of a single TCP flow computed with TDRM-UTEMA [46] and a memory of 0.5 s. The dark curves are without ABC, the light curves are with ABC. One-way delays of $D_b \in \{5, 25, 50\}$ ms are considered. These values correspond to BDPs of at least 8.5, 41.8, and packets, 83.5 packets. In contrast, the buffer is only 24 packets large and ABC traffic may not be able to fully utilize it. We observe that for $D_b = 5$ ms, TCP can utilize rather quickly the full bottleneck bandwidth, for $D_b = 25$ ms it



(a) User 0 and 1 send Poisson traffic with transmission rates R_t^0 and R_t^1. The throughputs T_0 of user 0 are given.

(b) User 0 and 1 send TCP traffic over f_0 and $f_1 = 1$ connections. The throughput ratios T_R of both users are given.

(c) User 0 sends Poisson traffic with rate R_t^0 and user 1 sends traffic over $f_1 = 1$ TCP connection. The throughput ratios T_R of both users are given.

Fig. 14. Impact of traffic type and configured unfairness on fairness with normal and fair activity meter (see Section III-E).

takes a few seconds, and for $D_b = 50$ ms it takes more than 10 s. Larger one-way delays lead to slower TCP rate increases. For $D_b = 25$ ms TCP's rate increase is faster with ABC than without ABC while for $D_b = 50$ ms it is vice-versa. Thus, we see TCP's rate increase difficulty with and without ABC. The performance depends on the specific setting. ABC's low drop threshold may be advantageous as it keeps the round-trip time short (in case of $D_b = 25$ ms), but ABC's low drop threshold may also be disadvantageous as it may provoke more severe packet loss (in case of $D_b = 50$ ms).

We visualize in Figure 13(b) the evolution of ABC's drop threshold over time in individual simulation runs for $M_{AA} \in \{0.1, 0.3, 1\}$ s and $D_b = 5$ ms. The drop thresholds are initially at $Q_{min} = 12$ and then converge to $Q_{base} = 20$ pkts. The convergence speed depends on M_{AA}. We explain the phenomenon as follows. The average activity A_{avg} at the activity AQM is initially zero. A starting TCP connection first exhibits a rather low measured rate for two reasons: the rate is first indeed low and activities are lower for users that have just started sending (see Section IV-G2). Therefore, measured activities are initially small and increase over time, the same holds for their average value A_{avg} at the activity meter. As the activity values increase, the average A_{avg} lags behind. This yields a positive difference $A - A_{avg}$ and leads together with $Q_{base} < Q_{max}$ to a low drop threshold for ABC (see Equation (10)). However, the drop threshold does not fall below the minimum threshold Q_{min}. The parameter study in Figure 13(b) shows that the activity averager memory M_{AA} influences the behavior of ABC's drop threshold: shorter values lead to faster adaptation of the drop threshold. However, the adaptation speed has mostly no impact on performance when M_{AA} is sufficiently small. We studied the time-dependent drop thresholds also for $D_b \in \{5, 25, 50\}$ ms, but they lead to similar curves (w/o figures).

H. Impact of Fair Activity Meter

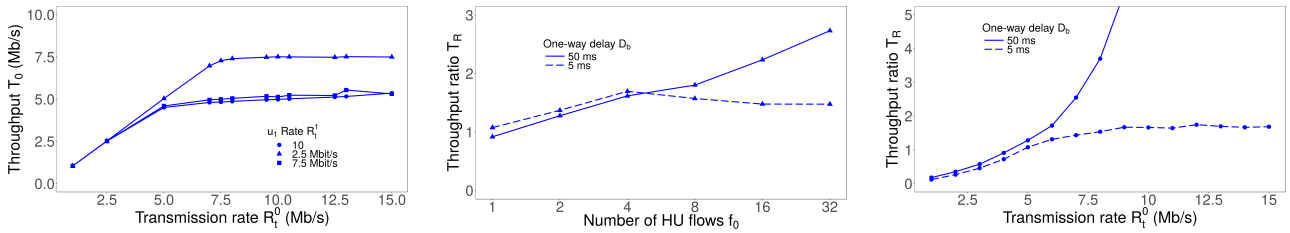
In Section III-E we have presented the fair activity meter as an alternative to the normal activity meter. It marks traffic with activity values such that equal-rate, least-activity subsets of two different aggregates have similar activities. As a result, these subsets should obtain similar treatment by the activity AQM. We evaluate the impact of the activity meter variant on fairness.

In the first experiment, user 0 and user 1 send Poisson traffic with different rates. We first consider the fair activity meter. Figure 14(a) shows that the throughput T_0 of user 0 converges for $R_t^1 \in \{5, 7.5\}$ Mb/s against slightly more than 5 Mb/s with increasing transmission rate R_t^0. Thus, user 0 obtains only slightly more than his fair share even if his transmission rate exceeds his fair share to a larger degree than user 1. In case of $R_t^1 = 2.5$ Mb/s, more capacity is available so that the throughput T_0 of user 0 converges against 7.5 Mb/s. With the normal activity meter, the traffic of the user with the lower transmission rate is prioritized over the traffic. As a result, user 0 obtains a lower throughput T_0 for large transmission rates R_t^0 with the normal activity meter. On the one hand, the fair activity meter improves fairness among users as it drops traffic such that users with different transmission rates achieve similar throughput while the normal activity meter leads to prioritization of light user traffic. On the other hand, this also reduces incentives for users to keep their transmission rates close to their fair shares.

In a second experiment, user 0 holds a variable number of f_0 TCP connections while user 1 holds only $f_1 = 1$ TCP connection. Figure 14(b) shows that ABC with the fair activity meter is able to protect the throughput of the light user against traffic of the heavy user, but only to a lower degree than ABC with the normal activity meter. Throughput ratios between 1.5 and 3 instead of between 1.0 and 1.5 are achieved. Thus, the fair activity meter reduces the fairness between heavy and light TCP users compared to the normal activity meter.

Finally, user 0 sends Poisson traffic at different transmission rates R_t^0 and user 1 sends traffic over $f_1 = 1$ TCP connection. Figure 14(c) shows that throughput ratios are between 2 and 3 with ABC and the fair activity meter instead of between 0 and 1.1 with the normal activity meter. Thus, the fair activity meter degrades the fairness for the responsive user.

In this work we concentrated on the normal activity meter as it enforces excellent fairness among responsive users at the expense of extremely impeding non-responsive users whose transmission rates exceed their fair shares. If that feature is not acceptable, the fair activity meter may be used instead as



(a) User 0 and user 1 send Poisson traffic with transmission rates R_t^0 and R_t^1. The throughput T_0 of user 0 is given.

(b) User 0 has multiple (f_0) TCP connections and user 1 has only $f_1 = 1$ TCP connection. The throughput ratios T_R of both users are given.

(c) User 0 sends Poisson traffic with rate R_t^0 and user 1 sends traffic over $f_1 = 1$ TCP connection. The throughput ratios T_R of both users are given.

Fig. 15. Impact of traffic type and configured unfairness on throughput or throughput ratio with CSFQ. The corresponding figures for ABC are Figure 3(a), Figure 4(a), and Figure 6(a).

it provides a similar ABC design without this feature but at the expense of reduced fairness.

Related control designs such as RFQ and PPV (see Section II) leverage marking schemes similar to the fair activity meter. Therefore, they suffer from the same drawback that they protect the throughput of light users to a lower degree against traffic of heavy users compared to ABC.

V. Comparison of ABC and CSFQ

CSFQ has been the first mechanism to enforce fair resource sharing in a core-stateless network and is most well-known. Therefore, we compare ABC with CSFQ. We explain our implementation and configuration of CSFQ and compare the fairness achieved with CSFQ with the one of ABC. Finally, we analyze why ABC leads to better results and compare the configuration requirements of ABC and CSFQ.

A. Configuration of CSFQ

The original source code for CSFQ is available at [54] for the ns2 simulation framework including the two minor amendments described in [34], [35]. We ported that code to ns3 in order to perform the same experiments as for ABC.

We used the following parameters for simulation which are recommended at [54]: $K = 100$ ms for measuring per-flow rates at edge nodes, $K_\alpha = 100$ ms for measuring aggregate rates $\hat{A}$ and $\hat{F}$ at edge and core nodes, $K_c = 100$ ms at core and edge nodes for the estimation interval of the estimated fair rate $\hat{\alpha}$. We utilize a bottleneck bandwidth of $C_b = 10$ Mb/s and a queue size of $Q_{max} = 24$ packets. The latter deviates from the original CSFQ evaluation and equals our evaluation of ABC. We deliberately chose a shorter queue size as overly large buffers contribute to bufferbloat. Moreover, typical switches exhibit a rather small queue sizes.

B. Non-Responsive Traffic

We first consider non-responsive traffic. Figure 15(a) shows that with CSFQ, the heavy and light user share the bandwidth very fairly. In contrast, with ABC heavy users achieve a significantly lower throughput with the normal activity meter (see Figure 3(a)) and a slightly too large throughput with fair activity meter (see Figure 14(a)).

C. Responsive Traffic

We study now a heavy and a light TCP user. The heavy user has a different number f_0 of TCP connections while the light user has only a single one. Figure 15(b) illustrates that CSFQ can enforce a throughput ratio between 1.4 and 1.6 for different number of flows f_0 and a one-way delay of $D_b = 5$ ms while ABC achieves values close to 1.0 (see Figure 4(a)). However, for larger one-way delay of $D_b = 50$ ms, the throughput ratio with CSFQ increases from 1.0 to 2.8 for an increasing number of flows f_0 of the heavy user. In contrast, ABC keeps the throughput ratio below 1.5 under these conditions.

D. Coexistence of Non-Responsive and Responsive Traffic

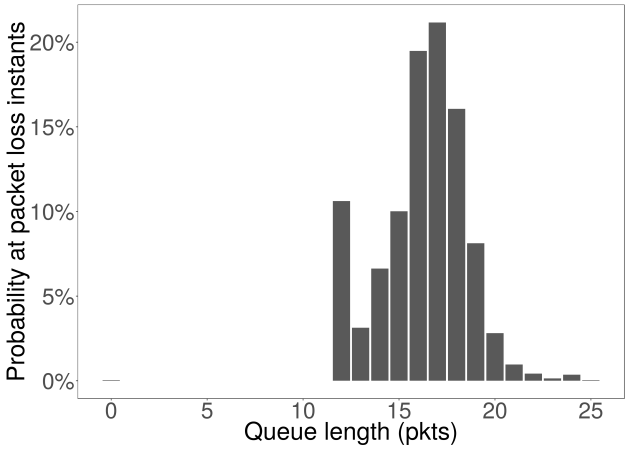
Finally, we investigate the coexistence of a heavy user sending Poisson traffic at rate R_t^0 and a light user with a single TCP flow. According to Figure 15(c) the throughput ratio increases with CSFQ from 0 to 1 when the transmission rate R_t^0 of the heavy user linearly increases from 0 to 5 Mb/s. We observe the same phenomenon with ABC in Figure 6(a). For larger transmission rates R_t^0, CSFQ can enforce a relative throughput of between 1.2 and 1.8 for small one-way delay of $D_b = 5$ ms. However, for a larger one-way delay of $D_b = 50$ ms, the throughput ratio significantly increases with increasing transmission rate R_t^0. Thus, CSFQ cannot protect light users against heavy users in case of longer one-way delays.

E. Discussion and Analysis

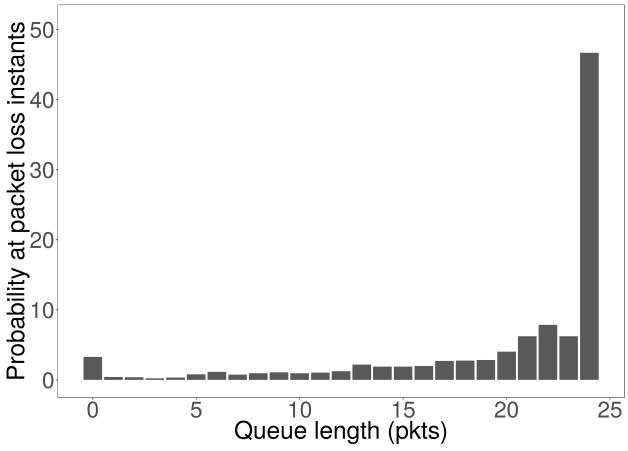
We compare CSFQ and ABC with regard to performance and configuration requirements.

1) Performance: We observed that CSFQ leads to larger throughput ratios than ABC in the presence TCP traffic and in particular in case of long one-way delays. To explain the effect, we visualize the distribution of the queue length in the presence of packet drops for ABC and CSFQ in Figures 16(a) and 16(b).

Figure 16(a) shows that with ABC, packets are dropped only when the queue length is at least Q_{min}. This is due to the definition of the activity AQM's drop threshold in Equation (10). Morover, packet drops occur only up to a queue



(a) ABC.



(b) CSFQ.

Fig. 16. Queue length distribution at packet drop instants. One heavy users sends $f_0 = 10$ TCP flows and $u_1 = 10$ light users send f_0 TCP flows. The one-way delay is $D_b = 50$ ms.

length of 23 packets, i.e., all drops are due to activity AQM drops and none due to buffer overflow. This is in line with our observations in Figure 10(c).

Figure 16(b) reveals that CSFQ has a significantly different drop behavior. Some packets are dropped when the queue is empty or if its length is rather small. This happens in spite of the implemented amendment in [35] which should prevent CSFQ dropping traffic when the queue length is small. This feature is added in Fig. 4 of [35] in the congested mode to check at the end of a measurement interval whether the queue is still sufficiently long. In that case CSFQ leaves the congested mode to prevent additional packet drops. However, that may be too late as the measurement interval takes $K_c = 100$ ms. As a result, CSFQ sometimes drops further packets although the queue has already sufficiently decreased. The solution is obvious: the algorithm in Fig. 3 of [35] must not drop packets when the queue is shorter than Q_{min}. We implemented this improvement, and observed that it avoids packet drops in the presence of short queues as desired, but it hardly improves throughput ratios.

In addition, we observe that CSFQ drops almost 50% of its packets when the queue is 24 packets long, i.e., these drops are due to buffer overflow. Such tail drops hit the traffic of both heavy and light users. This reduces CSFQ's drop differentiation between heavy and light users, which reduces CSFQ's ability to protect light users from heavy users under challenging conditions. As a result, CSFQ suffers from increased throughput ratios when TCP is transmitted and the one-way delay is only $D_b = 50$ ms. ABC leads to fairer resource sharing as the activity AQM is able to avoid tail drops with appropriate parameters Q_{min}, Q_{base}, and γ (see Section IV-F2).

2) Configuration Requirements: ABC is significantly simpler with respect to configuration and more adaptive than CSFQ. Like CSFQ, it requires per-flow rate measurement at edge nodes. However, at core nodes, CSFQ also requires rate measurement of the arrived and accepted traffic rate while ABC's activity AQM just computes a moving average of previously received activity values. Moreover, CSFQ needs to be configured with the bottleneck bandwidth C to control the estimated fair rate $\hat{\alpha}$ by $\hat{\alpha} = \hat{\alpha} \cdot C/\hat{F}$ during congestion phases [35, Fig. 4]. Therefore, CSFQ requires a constant and known bottleneck rate C. This is different for ABC which can therefore be also applied to transmission links with variable bandwidth.

VI. Conclusion

In this work we simplified activity-based congestion management (ABC) and extended it for traffic prioritization. We modified the activity meter at the edge nodes, which makes ABC easier to understand and configure. Likewise, we proposed a new activity AQM for forwarding nodes, which is more straightforward and easier to reason about. We added support for traffic prioritization and provided means to disincentivize high-priority transport if not needed and to avoid starvation of non-prioritized traffic.

We investigated fairness and queueing behavior with and without ABC under challenging conditions using packet-based simulation. We showed that ABC gives loss priority to users with lower activity so that users can maximize their throughput by sending at their fair rate. With ABC, users applying congestion control share the capacity of a bottleneck equally irrespective of their numbers of flows, which is different without ABC. Moreover, ABC protects the throughput of users sending congestion-controlled traffic against overload of users sending non-responsive traffic. This holds without and with prioritization of non-responsive traffic.

ABC's user-specific activity meters are configured with a reference rate. We proved that the reference rates of all users can be scaled by the same factor without changing the system behavior. Moreover, we showed that individual users can be granted larger capacity shares by scaling up their reference rates. We investigated the impact of ABC's activity AQM parameters and recommended values that lead both to good fairness and high link utilization. Both activity

meters in edge nodes and activity averagers in forwarding nodes require memories. We illustrated their impact and gave recommendations. Moreover, we showed that ABC reduces the time needed for small uploads, which improves the quality of experience for interactive applications.

Activity meters may be designed differently. We proposed a normal activity meter and a fair activity meter. While the fair activity meter improves the fairness for scenarios with only non-responsive traffic, the normal activity meter provides better fairness for responsive sources. The fair activity meter resembles the way how the similar control approaches RFQ and PPV perform traffic marking. We also showed that CSFQ, another similar approach, cannot provide the same fairness as ABC as it cannot efficiently avoid tail drops.

We recently implemented a prototype of ABC leveraging a programmable data plane and P4 [49] to demonstrate its technical viability. Future work should study use cases for which we see particular potential in data centers and in the wireline part of mobile networks.

Acknowledgements

The authors thank Bob Briscoe, David Wagner, and Habib Mostafaei for their helpful comments.

References

[1] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "5G Backhaul Challenges and Emerging Research Directions: A Survey," *IEEE Access*, vol. 4, pp. 1743 – 1766, Apr. 2016.

[2] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G Wireless Backhaul Networks: Challenges and Research Advances," *IEEE Network Magazine*, vol. 28, no. 6, pp. 6 – 11, Nov. 2014.

[3] B. Briscoe Ed., R. Woundy Ed., and A. Cooper Ed., "RFC6789: Congestion Exposure (ConEx) Concepts and Use Cases," Dec. 2012.

[4] Broadband Internet Technical Advisory Group (BITAG), "Real-Time Network Management of Internet Congestion," Broadband Internet Technical Advisory Group (BITAG), Tech. Rep., Oct. 2013.

[5] Sandvine Incorporated ULC, "Network Congestion Management: Considerations and Techniques – An Industry Whitepaper," Sandvine Incorporated ULC, Tech. Rep., Oct. 2015.

[6] IETF Working Group on RTP Media Congestion Avoidance Techniques (RMCAT), "Description of the Working Group," http://tools.ietf.org/wg/rmcat/charters, 2012.

[7] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "RFC6817: Low Extra Delay Background Transport (LEDBAT)," Dec. 2012.

[8] R. Pan, P. Natarajan, F. Baker, and G. White, "RFC8033: Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem," https://tools.ietf.org/html/rfc8033, Feb. 2017.

[9] K. Nichols, V. Jacobson, A. McGregor, Ed., and J. Iyengar, Ed., "RFC8289: Controlled Delay Active Queue Management," https://tools.ietf.org/html/rfc8289, Jan. 2018.

[10] M. Mathis and B. Briscoe, "RFC7713: Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements," Dec. 2015.

[11] D. Kutscher, F. Mir, R. Winter, S. Krishnan, Y. Zhang, and C. J. Bernados, "RFC7778: Mobile Communication Congestion Exposure Scenario," Mar. 2016.

[12] B. Briscoe, "Initial Congestion Exposure (ConEx) Deployment Examples," http://tools.ietf.org/html/draft-briscoe-conex-initial-deploy, Jan. 2012.

[13] B. Briscoe and M. Sridharan, "Network Performance Isolation in Data Centres using Congestion Policing," http://tools.ietf.org/html/draft-briscoe-conex-data-centre, Feb. 2014.

[14] M. Menth and N. Zeitler, "Activity-Based Congestion Management for Fair Bandwidth Sharing in Trusted Packet Networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Istanbul, Turkey, 2016.

[15] C. Bastian, T. Klieber, J. Livingood, J. Mills, and R. Woundy, "RFC6057: Comcast's Protocol-Agnostic Congestion Management System," Dec. 2010.

[16] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," in *ACM SIGCOMM*, 1989.

[17] M. Menth, M. Mehl, and S. Veith, "Fairness Enforcement for Multiple TCP Users through Deficit Round Robin with Limited Deficit Savings (DRR-LDS)," in *GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB)*, Erlangen, Germany, Feb. 2018.

[18] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the Data Center Network," in *USENIX Syposium on Networked Systems Design & Implementation (NSDI)*, 2011.

[19] R. Adams, "Active Queue Management: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1425–1476, 2013.

[20] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[21] K. Nichols and V. Jacobson, "Controlling Queue Delay," *ACM Queue*, vol. 10, no. 5, May 2012.

[22] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem," in *IEEE Workshop on High Performance Switching and Routing (HPSR)*, 2013.

[23] R. Pan, B. Prabhakar, and K. Psounis, "CHOKEs: a Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation," in *IEEE Infocom*, Tel Aviv, Israel, 2000.

[24] K. Ramakrishnan, S. Floyd, and D. Black, "RFC3168: The Addition of Explicit Congestion Notification (ECN) to IP," Sep. 2001.

[25] B. Briscoe, "Flow Rate Fairness: Dismantling a Religion," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, Apr. 2007.

[26] B. Briscoe, A. Jacquet, C. di Cairano-Gilfedder, A. Salvatori, A. Soppera, and M. Koyabe, "Policing Congestion Response in an Internetwork using Re-feedback," in *ACM SIGCOMM*, Portland, OR, Aug. 2005.

[27] A. Jacquet, B. Briscoe, and T. Moncaster, "Policing Freedom to Use the Internet Resource Pool," in *Re-Architecting the Internet (ReArch)*, Madrid, Spain, Dec. 2008.

[28] B. Briscoe, "Re-feedback: Freedom with Accountability for Causing Congestion in a Connectionless Internetwork," PhD thesis, Department of Computer Science, University College London, 2009.

[29] M. Kuehlewind, Ed. and R. Scheffenegger, "RFC7786: TCP Modifications for Congestion Exposure (ConEx)," May 2016.

[30] D. Wagner, "Congestion Policing Queues – A New Approach to Managing Bandwidth Sharing at Bottlenecks," in *International Conference on Network and Services Management (CNSM)*, 2014.

[31] S. Baillargeon and I. Johansson, "ConEx Lite for Mobile Networks," in *Capacity Sharing Workshop (CSWS)*, 2014.

[32] M. Menth and S. Veith, "Active Queue Management Based on Congestion Policing (CP-AQM)," in *GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB)*, Erlangen, Germany, Feb. 2018.

[33] M. Menth, B. Briscoe, and T. Tsou, "Pre-Congestion Notification (PCN) – New QoS Support for Differentiated Services IP Networks," *IEEE Communications Magazine*, vol. 50, no. 3, Mar. 2012.

[34] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *ACM SIGCOMM*, Vancouver, B.C., 1998.

[35] ——, "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High-Speed Networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, Feb. 2003.

[36] H.-T. Ngin and C.-K. Tham, "A Control-Theoretical Approach to Achieving Fair Bandwidth Allocations in Core-Stateless Networks," in *IEEE International Workshop on Quality of Service (IWQoS)*, Karlsruhe, Germany, 2001.

[37] C. Pelsser, , and S. De Cnodder, "Improvements to Core Stateless Fair Queueing," in *IFIP/IEEE International Workshop on Protocols for High-Speed Networks (PfHSN)*, Berlin, Germany, 2002.

[38] J. Kaur and H. M. Vin, "Core-stateless Guaranteed Throughput Networks," in *IEEE Infocom*, 2003.

[39] S. Shenker, C. Patridge, and R. Guerin, "RFC2212: Specification of Guaranteed Quality of Service," Sep. 1997.

[40] S. Blake, D. L. Black, M. A. Carlson, E. Davies, Z. Wang, and W. Weiss, "RFC2475: An Architecture for Differentiated Services," Dec. 1998.

[41] N. Sivasubramaniam and P. Senniappan, "Enhanced Core Stateless Fair Queuing with Multiple Queue Priority Scheduler," *The International Arab Journal of Information Technology*, vol. 11, no. 2, pp. 159 – 167, Mar. 2014.
[42] Z. Cao, Z. Wang, and E. Zegura, "Rainbow Fair Queueing: Fair Bandwidth Sharing Without Per-Flow State," in *IEEE Infocom*, 2000.
[43] Z. Cao, E. Zegura, and Z. Wang, "Rainbow Fair Queueing: Theory and Applications," *Computer Networks*, vol. 47, pp. 367–392, 2005.
[44] S. Nadas, Z. R. Turanyi, and S. Racz, "Per Packet Value: A Practical Concept for Network Resource Sharing," in *IEEE Globecom*, 2016.
[45] S. Laki, G. Gombos, S. Nadas, and Z. Turanyi, "Take Your Own Share of the PIE," in *ACM, IRTF & ISOC Applied Networking Research Workshop (ANRW)*, 2017.
[46] M. Menth and F. Hauser, "On Moving Averages, Histograms and Time-Dependent Rates for Online Measurement," in *ACM/SPEC International Conference on Performance Engineering (ICPE)*, L'Aquila, Italy, Apr. 2017.
[47] B. D. et al., "RFC3246: An Expedited Forwarding PHB (Per-Hop-Behavior)," Mar. 2002.
[48] M. Menth, M. Schmid, H. Heiß, and T. Reim, "MEDF - A Simple Scheduling Algorithm for Two Real-Time Transport Service Classes with Application in the UTRAN," in *IEEE Infocom*, San Francisco, USA, Mar. 2003.
[49] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming Protocol-Independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, 2014.
[50] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren, "Cloud Control with Distributed Rate Limiting," in *ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.
[51] D. Kutscher, F. Mir, R. Winter, S. Krishnan, Y. Zhang, and C. J. Bernados, "Mobile Communication Congestion Exposure Scenario," http://tools.ietf.org/html/draft-ietf-conex-mobile, Oct. 2015.
[52] ns-3 Consortium, "ns-3," http://www.nsnam.org/, Nov. 2014.
[53] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," DEC, Tech. Rep. Research Report TR-301, Sep. 1984.
[54] Ion Stoica, "CSFQ: Core-Stateless Fair Queueing," https://people.eecs.berkeley.edu/~istoica/csfq/, 1999.