Activity-Based Congestion Management for Fair Bandwidth Sharing in Trusted Packet Networks

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

***Abstract*—Congestion management detects congestion in a network and resolves it, e.g., by dropping or marking packets. A challenge is to drop or mark the right packets if fair capacity sharing is desired, especially if a few heavy users monopolize the bandwidth, e.g., by opening many flows or using non-responsive transport protocols. To this end, we propose activity-based congestion management (ABC). Users are assigned reference rates and their traffic is equipped with activity information. The activity indicates by which factor the transmission rate of a user exceeds his reference rate. ABC leverages active queue management (AQM) in routers or switches and uses the packets' activity information to adapt the drop or mark probabilities of the AQM. ABC is scalable as switches do not require user states, multiple queues, or signalling.**

We investigated ABC by means of simulation under conditions where a heavy user wants to monopolize a bottleneck's bandwidth. ABC provides an ecosystem where users with non-responsive constant-bitrate (CBR) traffic can maximize their throughput on a congested bottleneck link by adapting their sending rate to their fair share induced by their reference rate. Users with responsive TCP traffic obtain approximately fair capacity shares. Moreover, ABC protectes TCP traffic when competing with traffic from CBR users. We investigate the impact of system parameters and give recommendations for configuration.

I. Introduction

Congestion management detects and resolves congestion in a network. Simple forms effect that heavy users can monopolize a bottleneck link's bandwidth and degrade the quality of experience of competing light users. More sophisticated variants guarantee some fairness among users. Congestion management is useful, e.g., for multi-tenant data centers, mobile access and core networks, or Internet service provider (ISP) networks. In the latter, some back pressure is applied on certain users, traffic, or applications [1]–[3]. This helps ISPs to provide with limited resources better service to most customers, delay early reinvestments in transmission capacity, and save money.

In the recent years, there have been several efforts in particular in the IETF to improve congestion management for networks with respect to delay and fairness. New active queue management (AQM) algorithms are being standardized [4], [5]. Congestion exposure (ConEx) attempts to make congestion more visible in the IP layer to facilitate better-informed traffic engineering [1]. The driving application for ConEx is congestion policing that pursues more fairness among users in case of congestion [6]. One of its goals is to provide an ecosystem that incentivizes the application of more congestion-sensitive protocols such as LEDBAT [7]. Real-time transport protocol (RTP) media congestion avoidance techniques (RMCAT) add congestion control to realtime flows.

In this work, we introduce activity-based congestion management (ABC) to improve fairness among users in case of congestion. ABC defines traffic aggregates – based on user, interface, virtual machines, or other criteria – and assigns reference rates to them. In the following, we talk only about users for the sake of simplicity. A user's transmission rate is metered at the network edge and packets are equipped with activity information. AQM mechanisms in routers or switches inside a network adapt their mark or drop probabilities for packets depending on the activity coded in these packets. We show that ABC creates an ecosystem where users with constant bitrate (CBR) traffic can maximize their throughput by sending at their fair rate. Users with TCP traffic fairly share the capacity of a bottleneck link even if they hold a different number of flows. And TCP traffic is protected against overload by users sending non-responsive CBR traffic. Furthermore, ABC can be configured that some users obtain larger capacity shares and short transactions from light users are expedited.

ABC may be applied for similar use cases as ConEx-based congestion policing for which several preferred use cases have been proposed: residential access networks [8], mobile communication networks [9], and data center networks [10]. However, the focus of this paper is on the basic ABC mechanism, not on its adaptation to use cases. Note that ABC's objective is different from traditional AQMs. However, it may be combined with existing AQMs to provide both fairness and low delay which is not covered in this study.

The paper is structured as follows. Section II gives an overview of congestion management approaches. In Section III we propose the ABC design. The simulation results in Section IV show ABC's performance properties mentioned above and address configuration issues. Section V draws conclusions and gives an outlook on future work.

II. Related Work

An excellent and extensive overview of congestion management is compiled in [2]. 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.

978-1-5090-0223-8/16/$31.00 © 2016 IEEE
IEEE/IFIP Network Operations and Management Symposium (NOMS 2016), April 2016, Istanbul, Turkey

The whitepaper in [3] 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 [11] 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 possibly suffer from lower throughput and longer delays.

Scheduling mechanisms such as weighted fair queueing, possibly approximated by deficit round robin (DRR) [12], reduce a heavy user's throughput just to his fair share and only when needed. However, they are more complex 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 on potential bottlenecks.

Seawall [13] 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 [14] gives a comprehensive overview of the large set of existing mechanisms. RED [15] was one of the first AQMs and is implemented on many routers. CoDel [16] and PIE [17] are currently discussed and further developed in the IETF AQM working group to allow temporary bursts but to avoid a standing queue and bufferbloat. ABC leverages AQM mechanisms that drop packets with some probability. An example is PIE. Other AQM mechanisms, e.g., CoDel, work deterministically. Another AQM [18] 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) [19], 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 upon detection of packet loss.

Briscoe argued that per-flow fairness is not the right objective in the Internet [20] and proposed ReFeedback and congestion policing (CP) to implement per-user fairness [21]–[23]. 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 [24]. 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 [25].

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 [1] ConEx is qualitatively compared with existing congestion management approaches, and use cases are discussed which are further elaborated in [8]–[10]. There is only little insight into the performance of ConEx-based CP. Wagner [26] 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 [27]. Instead of requiring users to apply ConEx, traffic is tunneled and CP is performed using congestion feedback from the tunnel.

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 conges-

tion 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.

Core-stateless fair queueing (CSFQ) [28] resembles ABC in that edge nodes of a network add rate information to packets. Core routers measure the traffic arrival rate on a link and relate it to the known link bandwidth to determine congestion. The rate information contained in packets helps to find packet-specific drop probabilities to achieve max-min fairness among flows. In contrast, ABC leverages instantaneous queue lengths instead of rate measurements to detect congestion and determine drop probabilities. This is simpler and offers the perspective to combat bufferbloat in combination with appropriate AQMs and to cope with variable bandwidth.

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 [29]. While ABC meters traffic at the edge and drops packets inside a network, PCN meters traffic inside the network and drops packets from non-admitted or torn-down high-priority flows at the network edge. Thus, although ABC and PCN look similar at first sight, they are significantly different regarding operation and objectives.

III. ABC Design

We first give an overview of ABC. Then, we introduce the activity meter and explain how AQM probabilities are adapted. Finally, deployment aspects are discussed.

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 record it in their packets. Modified AQMs may drop/mark traffic in case of congestion on all links within an ABC domain. Such ABC-AQMs leverage only the packets' activity to preferentially drop/mark traffic from more active users. Therefore, ABC operation does not require per-flow or per-user state or signalling in the core network.

ABC assigns a reference rate R_r to the activity meter of any user (or other traffic aggregate) within an ABC domain. Activity meters determine the users' activity at every packet arrival. The activity is essentially the logarithmic value of the factor by which a user's transmission rate R_t exceeds his reference rate R_r. An ABC-AQM accounts for the average activity A_{avg} of received packets. It adapts the conventional AQM probability for dropping/marking a packet using the difference between a packet's activity and the observed average activity A_{avg} $(A - A_{avg})$. Essentially, drop/mark probabilities are increased for packets with activity values higher than A_{avg} and decreased for packets with activity values lower than A_{avg}.

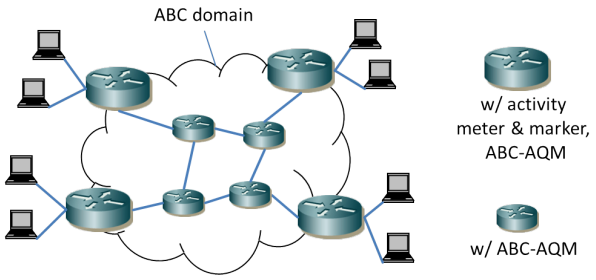

Fig. 1. Activity metering and marking is performed on the network edge only while ABC-AQM may be applied on all interfaces within an ABC domain.

B. Activity Meter

An activity meter is applied per user. It consists of a token bucket with a bucket size B and whose fill state F is increased by the user's reference rate R_r over time. The fill state F is decreased by the metered traffic and can become negative in contrast to conventional token buckets. If a packet of length L arrives at time t_{now}, the tokens arrived since the last packet arrival t_{last} are added to the fill state by

$$F = \min(B, F + R_r \cdot (t_{now} - t_{last})) \tag{1}$$

and t_{last} is updated with t_{now}. Then, the activity A is computed using the activity inertia I for scaling purposes:

$$A = \begin{cases} 0 & F \geq 0 \\ \frac{-F}{R_r \cdot I} & F < 0 \end{cases}. \tag{2}$$

The activity A is recorded in the packet's header. Then, the fill state F is decreased by

$$F = F - L \cdot 2^{-A}. \tag{3}$$

Both B and I are configured values. The bucket size B is given in bytes and expresses a burst allowance. We use a default value of $B = 0$ KB. An activity meter with a full bucket disregards at least this traffic quantity before indicating that the transmission rate R_t exceeds the reference rate R_r. The activity inertia I is given in seconds and controls how fast the activity adapts to changed transmission rates in a similar way as the length of a measurement window controls how fast measured rates can change over time.

We briefly summarize some properties of the proposed activity meter without further proof as the focus of this work is on the overall ABC mechanism which may use alternate activity meters. For a user with CBR traffic and a transmission rate $R_t > R_r$, the metered activity converges to $\log_2(R_t/R_r)$. The condition $R_r \cdot I > L$ should be respected so that the metered activity is sufficiently smooth. In our simulations, we take $I = 0.6$ s because this values allows to use small reference rates of $R_r = 0.02$ Mb/s without the risk of oscillating activity values.

C. Adaptation of AQM Probabilities

ABC adapts AQM probabilities such that packets from users with a larger or lower activity value than the average A_{avg} face larger or lower mark/drop probabilities. To that end, the AQM is extended to average activity values A of received packets by an exponentially weighted moving average (EWMA) as follows:

$$A_{avg} = w_A \cdot A_{avg} + (1 - w_A) \cdot A. \quad (4)$$

The activity weight w_A is a number between 0 and 1 and controls how quickly the average A_{avg} adapts to changed activity values. We use a default value of $w_A = 0.99$. The AQM probability p is adapted by the following equation:

$$p_A = p^{2^{-\gamma \cdot (A - A_{avg})}} \quad (5)$$

The differentiation factor γ controls the impact of the activity difference $(A - A_{avg})$ on the deviation of the modified drop probability p_A from the AQM probability p. Positive values of γ increase p_A for positive differences, which is desired. A value of $\gamma = 0$ avoids differentiation and turns off ABC. Negative differentiation factors lead to undesired adaptation results. We use a default value of $\gamma = 3$.

D. Deployment Aspects

ABC requires the following parameters: reference rate R_r, burst allowance B, activity inertia I per user, and differentiation factor γ and activity weight w_A per AQM on bottleneck links. They do not need to be the same network-wide, in particular users may be configured with different reference rates R_r for service differentiation. However, they need to be set consistently for meaningful operation. We study their impact in the next section. ABC does not require a special AQM. The AQM just needs to work with drop/mark probabilities that can be modified by ABC. The activity information may be coded, e.g., into IPv6 extension headers or in additional headers below IP. The latter seems doable in OpenFlow-based networks. Switches need to evaluate the activity information in packets, average over them, and adapt drop/mark probabilities of their AQMs accordingly. Fraudulent users may set too low activity values. Therefore, we propose the use of ABC only for trusted networks where the assignment of activities and the transport can be controlled by the network operator.

ABC requires any upstream traffic of a user to be activity-metered and equipped with activity information, which may be done in a single location close to the source. If ABC should be applied to downstream traffic, the user's traffic needs to be metered at possibly many network ingresses by a distributed activity meter. This seems feasible for two reasons. First, distributed policing has been demonstrated in [30]. Second, activity metering yields equal results when a flow is load-balanced over two meters that are configured with half the reference rate. Further details need to be discussed in the context of specific use cases which may be similar to those of ConEx-based CP [8]–[10].

IV. Results

In this section, we investigate bandwidth sharing with ABC using stochastic discrete-event simulation. After explanation of the simulation methodology, we first illustrate how competing constant-bitrate (CBR) flows share bandwidth with and without ABC. Then, we show that ABC protects TCP flows against non-responsive CBR flows. Finally, we demonstrate the ability of ABC to enforce per-user fairness for TCP traffic for various ABC configuration parameters and networking conditions.

A. Simulation Methodology

1) Simulator and Traffic Sources: We performed simulations with INET 2.4.0 [31] in the OMNet++ network simulation framework 4.4.1 [32]. We used the Network Simulation Cradle 0.5.3 [33] to model saturated TCP sources. It facilitates the application of real world network stacks from the Linux kernel for which we took version 2.6.29. We used TCP Reno for our study with and without ECN [19]. We work with a maximum transfer unit of MTU=1500 bytes on layer 2. For experiments with non-responsive traffic we apply CBR UDP sources with maximum packet size.

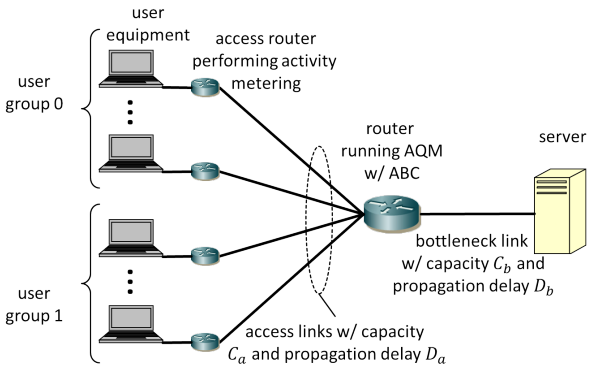

Fig. 2. Simulation setup.

2) Network and AQM: We simulate the scenario depicted in Figure 2. Multiple users communicate with a server via an access router, an access link, and a bottleneck link, yielding a one-sided dumbbell topology. They are divided into a user group 0 (UG_0) and a user group 1 (UG_1). If the experiments require a distinction between heavy and light users, the heavy users are grouped in UG_0 and the light users in UG_1. The user groups have u_i users which are all configured with the same reference rate R_r^i. All users of a user group communicate in the same way with the server. 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 CBR traffic at a transmission rate of R_t^i. 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 $2 \cdot (D_a + D_b)$.

The bottleneck link has a simple AQM algorithm that drops or marks packets with a probability that depends on the current

queue size Q given in packets and excluding the newly arrived packet. The probability function is determined as follows:

$$p(Q) = \begin{cases} 0 & Q \leq Q_0 \\ \frac{Q-Q_0}{Q_1-Q_0} \cdot p_1 & Q_0 < Q \leq Q_1 \\ p_1 + \frac{Q-Q_1}{Q_2-Q_1} \cdot (1-p_1) & Q_1 < Q \leq Q_2 \\ 1 & Q_2 < Q \end{cases}. \quad (6)$$

It has 4 parameters: thresholds Q_0, Q_1, Q_2, and probability p_1. Unlike RED, it does not average queue lengths Q [15]. We denote specific functions by "pf-Q_0-Q_1-Q_2-p_1". We choose this simple AQM to point out important AQM properties. However, this specific AQM is not essential for ABC and alternate approaches, e.g., PIE may be used. With the proposed AQM, the queue cannot exceed Q_2 packets in case of dropping so that this number of packets is a sufficient buffer size. With ECN-enabled TCP flows, packets are marked instead of dropped so that the queue size may increase significantly beyond Q_2. To minimize the likelihood of tail-drops, we generally use a buffer capacity of $Q_{max} = 50$ packets. On the fast access links, we apply drop-tail queues also with a buffer size of 50 packets.

3) Experiment Setup and Performance Metrics: 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 PARAMETER VALUES FOR EXPERIMENTS.

link bandwidths C_a, C_b	100 Mb/s, 10 Mb/s
link delays D_a, D_b	0.1 ms, 5 ms
AQM probability function	pf-11-17-24-0.01
buffer size Q_{max}	50 pkts
reference rates R_r^0, R_r^1	0.05 Mb/s
inertia I, burst allowance B	0.6 s, 0 bytes
diff. factor γ, activity weight w_A	3, 0.99

ABC's intention is to share bandwidth in case of congestion proportionally to users' reference rates. We study to which degree this objective can be achieved when heavy users compete with light users under various networking conditions.

Without ABC, saturated TCP flows, i.e., flows that always have data to send, share the bandwidth of a bottleneck link about equally, giving more throughput to users with more flows. We simulate heavy and light TCP users with f_0 and f_1 TCP flows ($f_0 \geq f_1$), respectively, yielding a configured unfairness of $U_c = \frac{f_0}{f_1}$. To study non-responsive traffic, we simulate a single user per user group UG_i with a single flow sending CBR traffic at rate R_t^i. 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 $U_c = \frac{R_t^0}{R_t^1}$.

The major intention of ABC is fair bandwidth sharing. 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. If $T_R = 1$ holds, the bandwidth is shared fairly among users in both groups. In case of $T_R > 1$, users in UG_0 receive higher throughput than users in UG_1, and in case of $T_R < 1$, users in UG_1 are advantaged. Maximizing per-user fairness means bringing T_R close to 1. The widely used fairness index of Jain [34] is at most 1 and does not reveal the advantaged user group.

4) Simulation Accuracy: For each experiment with TCP flows, we discarded a warmup phase of 15 s and collected simulation data over additional 10 s. We report mean values over 50 runs. The TCP sources were randomly started within the first 5 s of the simulation. In case of CBR users only, we report mean values over 100 runs with only 3 s warmup phase, 30 s data collection, and randomly start the traffic sources within the first second. We omit confidence intervals for the sake of better readability.

B. Performance of ABC with CBR Traffic

We show how two competing CBR users (user 0 and 1 in UG_0 and UG_1) share the bandwidth of a bottleneck link with and without ABC. We keep the transmission rate of user 1 fixed at $R_t^1 \in \{2.5, 5, 7.5\}$ Mb/s and study the throughput T_0 of user 0 depending on its transmission rate R_t^0. The dashed lines in Figure 3 show the results without ABC. The throughput T_0 of user 0 increases with increasing transmission rate R_t^0. Thus, there is no incentive for user 0 to respond to congestion.

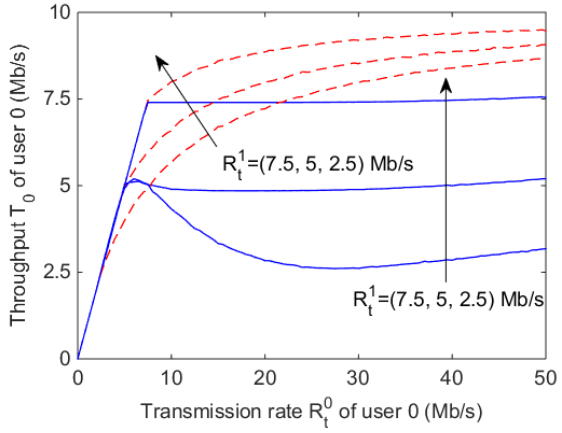

Fig. 3. Throughput T_0 of user 0 on a link with $C_b = 10$ Mb/s with (solid line) and without ABC (dashed line). ABC parameters are $R_r^0 = R_r^1 = 1$ Mb/s and $\gamma = 3$.

With ABC, user 0 and 1 are assigned equal reference rates of $R_r^{0,1} = 0.05$ Mb/s. The figure shows that the throughput of user 0 increases with increasing transmission rate R_t^0 up to a certain value, and then remains constant or even decreases with a further increase of transmission rate R_t^0. For $R_t^1 = 2.5$ Mb/s, user 0 can transmit up to 7.5 Mb/s but not more so that the throughput of user 1 ($T_1 = 2.5$ Mb/s) is hardly impacted by user 0. As user 1 does not exceed his fair share of 5 Mb/s while user 0 does so for larger transmission rates, ABC preferentially drops packets of user 0 so that the traffic of user 1 is protected. Similar holds for $R_t^1 = 5$ Mb/s. This is different if user 1 also exceeds his fair share. With $R_t^1 = 7.5$ Mb/s, user 0 approximatively obtains his fair share of 5 Mb/s in a range

5 Mb/s < R_t^0 < 7.5 Mb/s, but then his throughput diminishes as his activity is larger than the one of user 1. Thus, user 0 can maximize his throughput T_0 by sending at an appropriate rate. We conclude that ABC gives incentives to users to apply congestion control in order to maximize their throughput.

C. Performance of ABC with TCP and CBR Traffic

We consider a single heavy user sending CBR UDP traffic at rate R_t^0, competing for the bandwidth of the bottleneck link with $u_1 = 10$ light users with a single TCP flow each. All users are assigned the same reference rate of $R_r = 0.05$ Mb/s. We study the throughput ratio of heavy and light users for various differentiation factors $\gamma \in \{0,2,3\}$ and for bottleneck delays $D_b \in \{5,50\}$ ms. Figure 4 compiles the results.

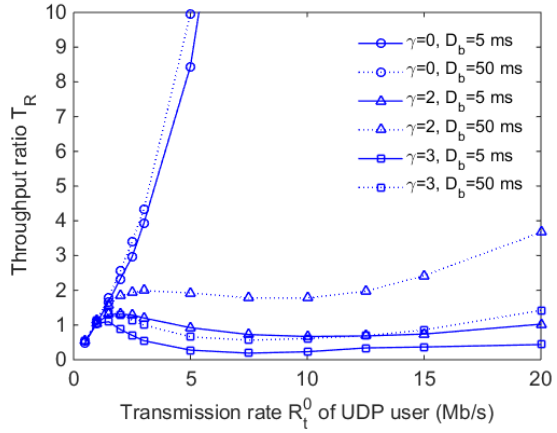

Fig. 4. Throughput ratio T_R of a CBR user and 10 TCP users.

With $\gamma = 0$, ABC is disabled. The figure shows that the throughput ratio T_R exceeds a value of 10 already for a transmission rate of $R_t^0 = 5$ Mb/s, i.e., the CBR user monopolizes the link and TCP users starve for larger transmission rates R_t^0 of the CBR user. Thus, bandwidth sharing among TCP and CBR users without ABC is highly unfair.

For our standard differentiation factor $\gamma = 3$, ABC is enabled and keeps the throughput ratio T_R between 0.6 and 1.5 for $D_b = 50$ ms and between 0.2 and 1.2 for a bottleneck delay of $D_b = 5$ ms even if the transmission rate R_t^0 is larger than the link bandwidth. Thus, ABC can well protect TCP users from greedy or unresponsive users.

A lower differentiation factor $\gamma = 2$ still limits the unfairness between CBR and TCP users compared to without ABC, but it leads to significantly larger throughput ratios T_R than $\gamma = 3$, especially for longer RTTs. Thus, $\gamma = 3$ is a good choice for ABC configuration. More evidence on the impact of γ will be given in the next section.

D. Performance of ABC with TCP Traffic

We investigate the impact of ABC parameters, AQM parameters, and ECN-enabled TCP flows. We show that ABC supports unequal capacity sharing and can significantly decrease upload times. In almost all conducted experiments, the bandwidth utilization of the bottleneck link is 100%. Thus, ABC shows no negative impact on link utilization.

1) Impact of ABC Parameters:

a) Impact of Differentiation Factor γ: We study the impact of γ for various combinations of number of users u_0/u_1 and bottleneck delays D_b as these parameters influence the congestion level on the bottleneck link. The configured unfairness is $U_c = \frac{f_0}{f_1} = \frac{10}{1} = \frac{90}{9} = 10$ in all considered scenarios. Table II shows that the throughput ratio T_R without ABC ($\gamma = 0$) is about $T_R \approx 10$. ABC with increasing γ decreases T_R significantly so that T_R may even fall below 1.0 for large values of γ. Exact numbers depend on the congestion level which is low for $D_b = 50$ ms and $u_0/u_1 = 1/10$ flows, and high for $D_b = 5$ ms and $u_0/u_1 = 9/90$ flows. We recommend $\gamma = 3$ because it assures even under challenging conditions ($u_0/u_1 = 1/10$ and $D_b = 50$ ms) a relatively large bandwidth share for light users ($T_R = 1.36$). In this case, the light users achieve lower throughput than the heavy user because they cannot fully exploit their relatively large fair capacity share due to the long RTT and their single TCP flow. Smaller bottleneck delays D_b effect that light users adapt their rate faster and can better exploit their fair share. More users u_0/u_1 effect that all users get a smaller fair share that light users with only a single flow can better exploit than large shares. Light users can obtain an even larger capacity share than heavy users in other scenarios ($T_R = 0.88$ for $u_0/u_1 = 9/90$ and $D_b = 50$ ms, $T_R = 0.91$ for $u_0/u_1 = 9/90$ and $D_b = 5$ ms). Larger differentiation factors γ fuel this effect so that we discourage their use. Throughput ratios below 1 are not ideal but we prefer them to throughput ratios larger than 1 under all conditions because a heavy user could reduce his transmission rate to improve his throughput. Thus, ABC gives incentives to heavy users to apply appropriate congestion controls and not to bypass their effect by increasing the number of flows. As TCP variants differ in aggressiveness, the experiments may yield slightly different results for other versions of TCP.

TABLE II
THROUGHPUT RATIO T_R DEPENDING ON DIFFERENTIATION FACTOR γ.

D_b (ms)	users u_0/u_1	differentiation factor γ					
		0	1	2	3	4	5
50	1/10	10.02	2.68	1.71	1.36	1.20	1.10
	9/90	10.20	1.51	1.03	0.88	0.82	0.80
5	1/10	10.14	1.90	1.26	1.03	0.94	0.89
	9/90	10.76	1.37	1.01	0.91	0.87	0.84

b) Impact of Activity Inertia I: We study the impact of the activity inertia $I \in [0.15, 4.8]$ for $\gamma = 3$ and the same congestion levels (D_b, u_0, u_1) as above. The inertia I has hardly any impact on the throughput ratio T_R (without figure).

c) Impact of Reference Rate R_r: We study the impact of the reference rate R_r for different numbers of heavy and light users u_0/u_1 and for different bottleneck link delays D_b. All users are configured with the same reference rate R_r. Figure 5 shows that heavy and light users receive about the same throughput as long as the reference rate is small

enough. If a critical rate is exceeded, heavy users receive up to 10 times more throughput than light users. This critical rate depends on the number of users. It is about 1 Mb/s for 1/10 users and 0.1 Mb/s for 9/90 users, and it is almost independent of the bottleneck delay D_b. We follow that the sum of the reference rates of all users should not exceed the bottleneck's bandwidth. This result can also be obtained from analytical studies. However, the numerical results for 1/10 users show that slight overbooking does not harm: $(u_0+u_1)\cdot R_r = 11\cdot 1$ Mb/s $= 11$ Mb/s > 10 Mb/s $= C_b$. Even though the critical rate depends on the number of users, ABC can be configured independently of that knowledge because it works well for smaller reference rates, too. Thus, an upper bound for a user's reference rate $R^+ = \frac{C_b}{u_{max}}$ can be computed as the fraction of the bottleneck bandwidth C_b and an upper bound on the number of active users u_{max} on that bottleneck resource.

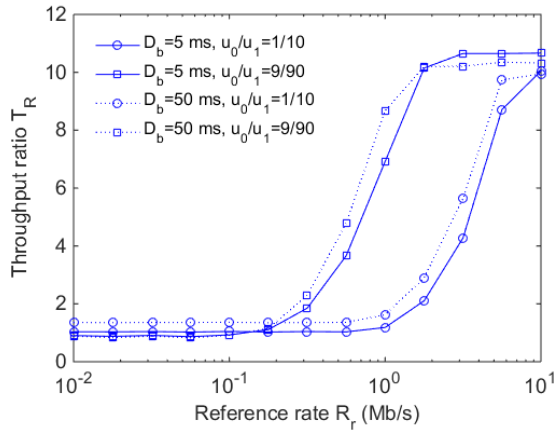

Fig. 5. Throughput ratio T_R depending on reference rate R_r.

d) Impact of Activity Weight w_A: We studied $w_A \in \{0.9, 0.99, 0.999, 0.9999\}$ for the same scenarios as above, but do not show results. The values $w_A \in \{0.9, 0.9999\}$ cause clearly increased throughput ratios under some conditions while $w_A \in \{0.99, 0.999\}$ lead to throughput ratios near 1 under all tested conditions. Therefore, we use $w_A = 0.99$ in our studies.

2) Impact of AQM Parameters: We show that ABC requires appropriate AQM probability functions to maximize per-user fairness. We discuss the probability functions listed in Table III first for the low-congestion scenario ($D_b = 50$ ms and $u_0/u_1 = 1/10$ users) and then for the high-congestion scenario ($D_b = 5$ ms and $u_0/u_1 = 9/90$ users).

With pf-11-23-24-0 a step function is implemented. Packets are accepted for queue occupancies 0 – 23 packets and dropped if the queue holds 24 packets which limits the maximum queue length. It leads to a throughput ratio of about $T_R = 10$. A step function essentially disables ABC as there is no queue occupancy level with a drop probability $0 < p < 1$. The latter is prerequisite for differentiation in Equation (5). The single-slope functions pf-11-{11,17,22}-24-0 accept all packets for occupancies 0 – {11,17,22} packets, drop all packets for occupancy 24, and drop packets with linearly increasing probability in between. Function pf-11-22-24-0 allows differentiation only for occupancy 23, which already reduces the throughput ratio from 9.96 to 4.30 compared with the step function. Functions pf-11-17-24-0 and pf-11-11-24-0 provide 6 and 12 occupancy levels for differentiation and reduce the throughput ratio further to 2.32 and 1.85, respectively. We observe for the latter two single-slope functions that average queue lengths Q_{avg} are so short that drop probabilities are mostly zero, which does not allow differentiation. The reason is that the drop probability of the first occupancy level with a non-trivial probability of a single-slope function is so large that the queue length stays mostly below that occupancy level in case of low congestion. Therefore, we consider double-slope functions (pf-11-17-24-{0.01,0.1}) whose probability first linearly increases to a small value and then again linearly to 1. They lead to longer average queue lengths being closer to the range where differentiation is possible. They cause the lowest throughput ratio of $T_R = 1.36$. We choose pf-11-17-24-0.01 as default for our experiments. More complex functions are possible. In particular PIE is interesting AQM candidate as it keeps queuing delay low. In contrast, CoDel is not compatible with ABC as its drop mechanism does not use probabilities.

TABLE III
IMPACT OF PROBABILITY FUNCTIONS.

D_b	50 ms		5 ms	
u_0/u_1	1/10		9/90	
prob. function	T_R	Q_{avg} (pkts)	T_R	Q_{avg} (pkts)
pf-11-23-24-0	9.96	17.28	10.09	23.27
pf-11-22-24-0	4.30	18.21	2.05	22.68
pf-11-17-24-0	2.32	15.21	0.92	20.00
pf-11-11-24-0	1.85	9.91	0.88	16.87
pf-11-17-24-0.1	1.38	11.36	0.90	19.60
pf-11-17-24-0.01	1.36	13.42	0.91	19.95

The high-congestion scenario ($D_b = 5$ ms, $u_0/u_1 = 9/90$) leads to clearly larger average queue lengths Q_{avg} so that the queue length is mostly in a range that allows differentiation of drop probabilities. This leads to lower throughput ratios T_R. The step function again disables ABC.

3) Impact of ECN-Capable Flows: We study ABC for ECN-capable flows whose packets are marked instead of dropped by the AQM. As the queue can grow significantly, we investigate queues with a buffer sizes of 24 and 50 packets. Almost any high-load occupancy level (occupancy 12–23) in the short queue yields a non-trivial drop probability while the long queue has additional occupancies 25–50 where any packet is marked or dropped without any possibility for differentiation. Throughput ratio and average queue lengths are compiled for this setting in Table IV including comparative results without ECN.

We observe that the throughput ratio T_R is larger for ECN and a buffer size of $Q_{max} = 24$ packets than without ECN, especially for long RTT. This can be explained as follows. Since packets of heavy users are rather marked than dropped,

TABLE IV
IMPACT OF ECN-ENABLED FLOWS.

scenario		$D_b = 50$ ms		$D_b = 5$ ms	
ECN	u_0/u_1	T_R	Q_{avg}	T_R	Q_{avg}
off	1/10	1.23	12.57	1.03	16.31
	3/30	1.05	16.31	0.93	17.66
	9/90	0.88	18.43	0.91	18.95
on $Q_{max} =$ 24 pkts	1/10	2.40	7.26	1.92	14.64
	3/30	3.58	11.69	1.81	18.49
	9/90	2.15	18.26	1.66	21.43
on $Q_{max} =$ 50 pkts	1/10	2.63	7.32	3.05	14.05
	3/30	5.98	13.53	3.68	21.61
	9/90	5.20	22.71	3.76	25.00

heavy users strongly contribute to high-load positions of the queue in case of congestion. Then, tail-drops become more likely hitting both heavy and light users. This diminishes the degree of differentiated treatment for heavy and light users and increases the throughput ratio due to the configured unfairness. As another consequence, the average queue length Q_{avg} may increase, especially in high-load scenarios. In case of long D_b and only a few users (1/10), we observe reduced queue lengths and diminished resource utilization around 90% for both the short and the long queue size. As the AQM marks packets instead of dropping them, the queue increases quickly to an occupancy level without differentiation. Therefore, light user flows are also hit by marked or dropped packets. This significantly throttles the overall traffic rate and the few users are not able to increase their rates fast enough to keep the pipe filled.

With a buffer size of $Q_{max} = 50$ packets, the throughput ratio increases even further to values between 3 and 4 or 5 and 6, especially for many flows. The average queue length Q_{avg} increases to a range where all packets are marked or dropped and differentiation based on activity values is no longer possible. Therefore, flows of heavy and light users are treated equally as long as the queue stays in this range, which increases the throughput ratio.

Thus, ABC does not work well with ECN under some conditions, in particular for long queue sizes. Therefore, we recommend that ABC-enabled AQMs disregard ECN, i.e., ECN-capable packets should be dropped instead of marked.

4) Unequal Capacity Shares: ABC is designed to support unequal capacity shares by configuration of appropriate reference rates. We investigate to what extent a privileged user can exploit this bandwidth share with a single TCP flow. We assume UG_0 holds only the privileged user with a single flow and being configured with a scaled reference rate of $R_r^0 = s \cdot R_r^1$. UG_1 holds multiple users with a single flow each. Table V shows for multiple congestion levels that the throughput ratio T_R increases with the scaling factor s. The growth is sublinear and depends on the congestion level. However, further experiments have shown that the privileged user can fully leverage his increased bandwidth share when using multiple flows.

5) Reduced Upload Times: ABC can reduce upload times for occasional transactions of moderate size. We model a single "probe" user who transmits a probe of 1 MB on application layer. Background traffic is produced by u_1 users. They send with f_1 saturated TCP flows each and have started transmission long before the probe user. Upload times for the probe are compiled in Table VI. Without ABC, the upload time significantly increases with the overall number of flows $(u_1 \cdot f_1 + 1)$ and it is longer for $D_b = 50$ ms than for $D_b = 5$ ms.

TABLE V
THROUGHPUT RATIO T_R FOR A PRIVILEGED USER WITH A SINGLE FLOW AND A SCALED REFERENCE RATE $R_r^0 = s \cdot R_r^1$.

D_b (ms)	no. users u_0/u_1	scaling factor s				
		1	2	4	8	16
50	1/1	1.01	1.55	2.47	3.78	4.30
	1/10	1.00	1.68	2.75	4.62	7.85
	1/100	1.01	1.85	3.03	5.07	7.39
5	1/1	1.00	1.88	3.25	5.95	10.45
	1/10	1.00	1.82	3.23	5.71	10.39
	1/100	1.01	2.00	3.78	5.98	10.91

TABLE VI
UPLOAD TIMES (S) WITH AND WITHOUT ABC.

D_b (ms)	ABC	B (MB)	$f_1 = 1$		$f_1 = 4$	
			$u_1 = 10$	$u_1 = 20$	$u_1 = 10$	$u_1 = 20$
50	no	n/a	20.3	29.2	47.7	85.2
	yes	0	9.6	17.9	12.3	21.7
		0.25	8.2	15.7	10.4	17.6
		0.5	6.1	10.7	8.3	14.0
		0.75	2.2	5.7	6.7	8.1
		1	2.4	2.7	3.3	5.4
		1.25	2.1	3.3	3.3	5.2
5	no	n/a	10.3	20.2	34.2	61.4
	yes	0	9.6	17.6	10.1	18.8
		0.25	6.9	13.7	8.3	14.0
		0.5	4.6	9.5	5.6	10.0
		0.75	1.3	5.4	4.0	6.4
		1	1.0	1.3	1.0	1.4
		1.25	1.0	1.0	1.0	1.0

We perform the same experiments with ABC while varying the burst allowance B for all users. Table VI shows that ABC significantly reduces the upload time for multiple flows per background user ($f_1 = 4$) *or* for $D_b = 50$ ms while its improvement a single flow per background user ($f_1 = 1$) *and* for $D_b = 5$ ms is comparatively low. For multiple flows ($f_1 = 4$), ABC reduces the upload time substantially because it partitions the bottleneck bandwidth equally among users. With a single user and a long delay of $D_b = 50$, the upload time is still clearly shorter than without ABC because the activity of the probe user remains lower than the one for background users for a while, leading to lower drop probabilities and improved throughput. Increasing the burst allowance B reduces the upload time even further until B exceeds the size of the probe plus protocol overhead and retransmitted packets. For very large B, the upload time is orders of magnitude shorter than without ABC. ABC essentially prioritizes the probe over other traffic as long as the burst allowance prevents the activity meter's fill state to go negative so that probe packets have an activity of zero. Therefore, this feature needs to be handled carefully. Appropriate values for the burst allowance B depend on the context.

V. Conclusion

Activity-based congestion management (ABC) enables users in an ABC domain to obtain a fair share of a bottleneck capacity in case of congestion. An activity meter at the network edge equips the user's traffic with activity information, and an extension of active queue management (AQM) in routers or switches inside the network leverages the packets' activity to adapt the AQM's loss/mark probability. ABC is scalable because it does not require per-user states or additional queues inside the network.

We simulated ABC with constant-bitrate (CBR) traffic, CBR and TCP traffic, and with TCP traffic only. ABC gives better treatment to flows not exceeding their fair share, incentivizing the use of congestion control algorithms. ABC protects TCP users against CBR traffic. We showed that with ABC, heavy and light users can obtain about equal bandwidth shares on a congested link and investigated configuration issues. ABC supports unequal capacity shares if desired and reduces upload times for transactions with moderate burst sizes significantly.

Future work should demonstrate the technical viability of ABC by implementation. ABC should be adapted to specific use cases and its benefits should be quantified in these contexts. The interplay of ABC with other congestion control algorithms and traffic needs further study. A combination of ABC with AQM algorithms like PIE seems feasible and may enforce both fairness among users and low latency.

Acknowledgements

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

References

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

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

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

[4] R. Pan, P. Natarajan, F. Baker, B. VerSteeg, M. Prabhu, C. Piglione, V. Subramanian, and G. White, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem," http://tools.ietf.org/html/draft-ietf-aqm-pie, Nov. 2015.

[5] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," http://tools.ietf.org/html/draft-ietf-aqm-codel, Dec. 2015.

[6] B. Briscoe, "Network Performance Isolation using Congestion Policing," http://tools.ietf.org/html/draft-briscoe-conex-policing, Feb. 2014.

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

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

[9] 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.

[10] 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.

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

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

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

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

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

[21] 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.

[22] 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.

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

[24] M. Kühlewind and R. Scheffenegger, "TCP Modifications for Congestion Exposure," http://tools.ietf.org/html/draft-ietf-conex-tcp-modifications, Oct. 2015.

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

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

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

[28] I. Stoica, S. Shenker, and H. Zhang, "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.

[29] 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.

[30] 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.

[31] A. Varga, "INET-2.4.0 released," http://inet.omnetpp.org/, Jun. 2014.

[32] ——, "OMNeT++ 4.4.1 released," http://www.omnetpp.org/, Oct. 2014.

[33] S. Wand, "Network Simulation Cradle," http://research.wand.net.nz/software/nsc.php, 2012.

[34] 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.