Activity-Based Congestion Management (ABC)

Michael Menth
University of Tuebingen
Tuebingen, Germany

Nikolas Zeitler
University of Tuebingen
Tuebingen, Germany

ABSTRACT

Congestion management detects congestion in a network and drops or marks packets to mitigate congestion. 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 their drop or mark probabilities. ABC is scalable as switches do not require user states, multiple queues, or signalling. Analytical and simulation studies demonstrate that ABC enables users to obtain fair capacity shares on a congested bottleneck that are approximatively proportional to their reference rates. This holds for TCP traffic, non-responsive traffic, or mixes thereof. We investigate the impact of system parameters and give recommendations for configuration. ABC provides an ecosystem that incentivizes the use of congestion control protocols and protects TCP traffic against overload through non-responsive traffic.

Keywords

Congestion management, per-user fairness, active queue management

1. INTRODUCTION

Internet service providers (ISP) apply congestion management techniques to provide the best experience for most of the applications, most of their users, most of the time based on available network resources. It is motivated by the observation that congestion often occurs due to a few heavy users monopolizing a network's bandwidth which impacts the quality of experience of most other users. Thus, some back pressure is applied on certain users, traffic or applications in case of congestion, to provide better quality for others [9, 10, 23]. This helps ISPs to provide better service with limited resources, 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. Congestion exposure (ConEx) attempts to make congestion more visible in the IP layer to facilitate better informed traffic engineering. The driving application is congestion policing that pursues more fairness among users in case of congestion. One of its goals is to provide an ecosystem that incentives the application of more congestion-sensitive protocols such as LEDBAT [24]. 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. It may be combined with AQM mechanisms to also minimize delay which is not covered by this study. ABC assigns reference rates to users, interfaces, or other traffic aggregates. In the following, we talk only about users for the sake of simplicity. A user's transmission rate is metered to equip his traffic with activity information. It is leveraged by AQM mechanisms in routers or switches to drop packets with increased or decreased probability in case of congestion. As a result, the available capacity can be shared fairly among users with respect to their reference rates. Furthermore, ABC expedites short transactions for light users and protects TCP traffic against non-responsive traffic. ABC may be applied in a similar way as congestion policing for which several preferred use cases have been proposed: residential access networks [6], mobile communication networks [16], and data center networks [8]. However, the focus of this paper is on the basic ABC mechanism, not on its adaptation to use cases.

The paper is structured as follows. Section 2 gives an overview of congestion management approaches. In Section 3 we propose the design of ABC including a token-bucket based meter for rate measurement. Section 4 gives insights into the behavior of the activity meter and ABC for constant bit rate (CBR) traffic based on analytical results. The simulation results in Section 5 show the effectiveness of ABC and address configuration issues. Section 6 draws conclusions and gives an outlook on future work.

2. RELATED WORK

An excellent and extensive overview of congestion management is compiled in [10]. 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 by ISPs are described.

The whitepaper in [23] proposes that congestion management should be applied only in the presence of real congestion and discusses several detection methods 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 very 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 [3] 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 robing (DRR) [11], 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 on potential bottlenecks for per-user fairness.

Seawall [25] 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 [1] gives a comprehensive overview of the large set of existing mechanisms. RED [12] was one of the first AQMs and is implemented on many routers. CoDel [18] and PIE [20] 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. Another AQM [19] 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) [22], ECN-enabled TCP senders mark packets appropriately and AQM mechanisms mark their packets as "congestion experienced" (CE) instead of dropping them. 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 [4] and proposed ReFeedback and congestion policing to implement per-user fairness [5, 7, 13]. 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. Modifications to TCP senders and receivers are required [15]. The network cannot trust that users insert sufficient ConEx marks into a packet stream. Therefore, per-flow audit near the receiver should compare CE and ConEx signals to detect cheating flows and sanction them [17].

We briefly explain ConEx-based congestion policing 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, the policer drops some of the user's traffic. The policer penalizes heavy users causing a lot congestion and saves light users whose ConEx-marked traffic rate does not exceed their congestion allowances. The objective of this differentiated treatment is fairer bandwidth sharing among users in case of congestion. In [9] ConEx is compared with existing congestion management approaches, and use cases are discussed which are further elaborated in [6, 8, 16]. There is only little insight into the performance of congestion exposure. Wagner [28] applied congestion policing to a single queue where traffic of different users is classified and separately policed before entering the queue. This approach requires per-user state so that the major advantage over scheduling is lost. ConEx Lite was developed for mobile networks in [2]. Instead of requiring users to apply ConEx, traffic is tunneled

and congestion policing is performed based on tunnel feedback.

ABC was inspired by ConEx but takes a different approach due to lessons learned. We have simulated ConEx-based congestion management and gained two insights. First, approximating per-user fairness is challenging with ConEx. 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 congestion policing we considered benefits from leveraging information about bottleneck bandwidths and the number of current users for system configuration.

3. 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.

3.1 Overview

ABC assigns a reference rate R_r to any user in the network. The activity of a user is the logarithmic value of the factor by which his transmission transmission rate R_t exceeds his reference rate R_r. It is calculated by an activity meter at every packet arrival and coded into the packet header. Potential bottleneck links run an appropriate AQM mechanism that marks/drops packets with some probability. It further accounts for the average activity A_{avg} of received packets. The AQM probability for dropping/marking a packet is adapted using the difference $(A - A_{avg})$ between a packet's activity and the observed average activity 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}. As a consequence, users on a common bottleneck link can share its capacity in the presence of congestion about proportionally to their reference rates R_r if they adapt their transmission rates R_t appropriately; otherwise they may receive lower throughput.

3.2 Activity Meter

The activity meter is a token bucket with a bucket size B 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 by

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

It 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. 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 quickly the activity adapts in response to rate changes and traffic bursts. We use default values of $B = 0$ KB and $I = 0.6$ s.

3.3 Adaptation of AQM Probabilities

The objective is to adapt AQM probabilities such that packets from users with a larger or lower activity value than the average 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:

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

For the activity weight w_A holds $0 < w_A < 1$ and it 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:

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

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 negative differentiation factors lead to undesired adaptation results. We use a default value of $\gamma = 3$.

3.4 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 sections. 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.

ABC requires any upload 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 downloads, 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 [21]. 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 congestion policing [6, 8, 16].

4. ANALYSIS

The results in this section are gained by a mathematical analysis considering constant bit rate (CBR) traffic. We first study the behavior of the activity meter. Then, we illustrate the capacity sharing result of ABC under various conditions. Finally, we explain why some implicit parameters have no impact on bandwidth sharing results and give recommendations for the configuration of reference rates.

4.1 Analysis of the Activity Meter

We consider the evolution of the activity of a user with a transmission rate R_t, reference rate R_r, burst allowance B, and activity inertia I. He starts transmitting CBR traffic time $t = 0$ s and stops at time $t = 5$ s. We model the evolution of the fill state F of the token bucket for each packet arrival. First, the fill state F is increased by the number of tokens arrived since the last packet arrival according to Equation (1). Then, the activity is calculated Equation (2). Finally, the packet is metered which updates the fill state F according to Equation (3). These steps are repeated which essentially simulates the evolution of the fill state F.

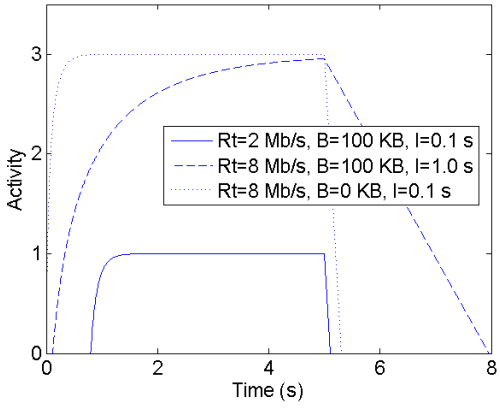

Figure 1: Evolution of activity values over time for a CBR user with reference rate $R_r = 1$ Mb/s.

Figure 1 shows the evolution of the metered activity of a CBR user with a reference rate $R_r = 1$ Mb/s, a transmission rate $R_t = 2$ Mb/s, a burst allowance $B = 100$ KB, and an inertia $I = 0.1$ s. The activity becomes positive when the fill state F falls below zero. This happens at time $t_A^+ = \frac{B}{R_t - R_r}$ for $R_t > R_r$. Thus, we get $t_A^+ = 0.8$ s for $B = 100$ KB and $R_t = 2$ Mb/s, $t_A^+ = 0.11$ s for $B = 100$ KB and $R_t = 8$ Mb/s, and $t_A^+ = 0$ for $B = 0$ KB as illustrated in the figure.

After time t_A^+, the activity increases quickly and slowly approaches a stationary activity A_s. When the stationary activity is reached, the token rate removed from the token bucket equals the reference rate R_r:

$$R_t \cdot 2^{-A_s} = R_r \tag{6}$$

which facilitates the computation of the stationary activity

$$A_s = \ln\left(\frac{R_t}{R_r}\right) / \ln(2) = \log_2\left(\frac{R_t}{R_r}\right). \tag{7}$$

Thus, the activity is the logarithmic value of the factor by which the transmission rate R_t exceeds the reference rate R_r. It depends on R_t and R_r but not on the activity inertia I. Figure 1 illustrates these findings: we observe $A_s = 1$ for $R_t = 2$ Mb/s and $A_s = 3$ for $R_t = 8$ Mb/s independently of the activity inertia I.

The user stops transmission at time $t = 5$ s. As the activity has converged to its stationary value A_s, the fill state is approximately $F = -R_r \cdot I \cdot A_s$ bytes large. From then on, the fill state linearly increases over time and the activity A linearly approaches zero. This process takes $\frac{R_r \cdot I \cdot A_s}{R_r} = I \cdot A_s$ time. Figure 1 shows that the activity returns to zero very quickly for $I = 0.1$ s when the user stops sending and it takes longer for $I = 1.0$ s.

In a similar way, the time after transmission start for the activity to come close to its stationary value, say $A_s - \varepsilon$, depends on the inertia I while the impact of the actual value A_s is rather low. This can be shown by lengthy elementary calculus. It is also illustrated in Figure 1 where the two curves with $I = 0.1$ have almost identical shape.

The activity inertia I influences the resistance of the activity to traffic bursts. We derive the burst size V needed to increase the activity of a user with a transmission rate R_t and stationary activity A_s by 1. With the burst's arrival, the activity A increases from A_s to $A_s + 1$. To that end, the fill state F must decrease by $R_r \cdot I$ bytes. As packets contribute only with a fraction between $2^{-(A_s+1)}$ and 2^{-A_s}, a burst size $2^{-(A_s+1)} \cdot V < I \cdot R_r \leq 2^{-A_s} \cdot V$ is needed. With Equation (6) this inequality can be transformed into $R_t \cdot I \leq V < 2 \cdot R_t \cdot I$. Thus, the burst size needed to increase the activity by 1 depends on the transmission rate R_t and the inertia I, but it is independent of the configured reference rate R_r.

The distance between fill states in the token bucket whose corresponding activities differ by 1 is $R_r \cdot I$. To ensure that a packet arrival does not cause a jump by 1 in activity, the activity inertia should be chosen such that $R_r \cdot I$ is at least a packet length L:

$$R_r \cdot I \geq L. \tag{8}$$

If a lower inertia is chosen, activity values may strongly oscillate even for CBR traffic that exceeds its reference rate

only moderately. Figure 2 illustrates that phenomenon for a small reference rate $R_r = 0.01$ Mb/s, a transmission rate $R_t = 0.02$ Mb/s, a transmission time of 5 s, and for various inertia values I. Only the dots represent activity values at packet arrivals, the lines just link them to indicate they belong together. The curve for an inertia of $I = 1.2$ s is still smooth as it respects $R_r \cdot I \leq 1500$ bytes, curves for lower inertia values tend to oscillate, in particular the activity values for the second packet are increased.

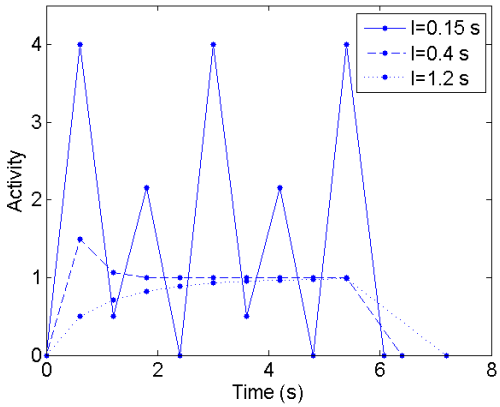

Figure 2: Evolution of activity values over time for a CBR user with reference rate $R_r = 0.01$ Mb/s and transmission rate $R_t = 0.02$ Mb/s.

4.2 Analysis of ABC

We investigate how two users sending CBR traffic share a bandwidth of $C_b = 10$ Mb/s on a common bottleneck link with and without ABC. First, we study the impact of the differentiation factor γ and demonstrate that unequal capacity sharing can intentionally be enforced. Then, we show that ABC protects a fair traffic share of users with lower activity against traffic of users with higher activity.

4.2.1 Impact of the Differentiation Factor γ

In this experiment, the transmission rate R_t^0 of user 0 varies and the transmission rate of user 1 is set to $R_t^1 = 7.5$ Mb/s. Without ABC, the bottleneck's capacity is shared proportionally to the transmission rates of both users ($R_t^0 : R_t^1$). The dashed curve in Figure 3 illustrates the throughput T_0 of user 0 under these conditions and depending on his transmission rate R_t^0. His throughput T_0 increases steadily with increasing transmission rate R_t^0. For small transmission rates, user 0 achieves only small throughput, and for large transmission rates, he monopolizes the bottleneck's link capacity.

We now analyze bandwidth sharing with ABC. The stationary activities of the users depend on their transmission rates R_t^i and reference rates R_r^i, and can be computed with Equation (7). On the bottleneck link, the average activity $A_{avg} = (\sum_{i\in\{0,1\}} A_s^i \cdot R_t^i)/(\sum_{i\in\{0,1\}} R_t^i)$ is computed by weighting the users' activities with their transmission rates

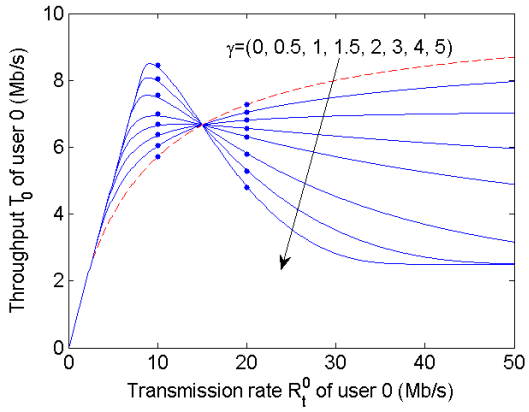

Figure 3: Throughput T_0 of user 0 with and without ABC on a link with $C_b = 10$ Mb/s and a transmission rate of user 1 of $R_t^1 = 7.5$ Mb/s. ABC parameters are $R_r^0 = 2$ Mb/s and $R_r^1 = 1$ Mb/s. The dots show simulation results.

R_t^i. Based on the average activity A_{avg} and the users' activities, the the average drop probabilities p_A^i for both users are adapted with Equation (5). However, this equation requires the AQM drop probability p. We choose it such that the sum of the throughput values T_i equals the bandwidth C_b of the bottleneck link: $\sum_{i\in\{0,1\}} T_i = \sum_{i\in\{0,1\}} (1 - p_A^i) \cdot R_t^i = C_b$. We calculate p by an iterative approximation which also yields the desired throughput values T_i for $i \in \{0,1\}$. In Section 5.2 we validate analytical results with data from a packet-based simulation and show that the analysis provides very accurate results.

To illustrate the effect of unequal reference rates, we set the reference rate of user 1 to $R_r^1 = 1$ Mb/s, and the one of user 0 to $R_r^0 = 2$ Mb/s. Thus, a bandwidth partition of $\frac{T_0}{T_1} = \frac{R_r^0}{R_r^1} = \frac{2}{1}$ is intended so that user 0 should be able to achieve a throughput of $T_0 = 6.67$ Mb/s. The solid curves in Figure 3 show the throughput T_0 of user 0 depending on his transmission rate R_t^0 for various differentiation factors γ. ABC effects that user 0 receives a larger bandwidth share than without ABC already for low transmission rates and that he does not monopolize the link for large transmission rates R_t^0. The curves clearly depend on the differentiation factor γ. As a differentiation factor of $\gamma = 0$ does not differentiate drop probabilities, its results equal those without ABC. Small values of $\gamma \in \{0.5, 1\}$ lead to a moderate growth of T_0 first, and effect that T_0 continues growing even for large values of R_t^0. Larger values of $\gamma \in \{1.5, 2, 3, 4, 5\}$ lead to larger growth of T_0 for small values of R_t^0, effect an overshoot of the intended share, and cause that T_0 eventually converges for large R_t^0 to the rate that is unused by user 1 which is $C_b - R_t^1 = 2.5$ Mb/s in the experiment. Thus, user 0 cannot monopolize the bottleneck's bandwidth by increasing his transmission rate, his traffic is rather deprioritized if he exceeds his reference rate more than the competing users.

Thus, he can maximize his throughput T_0 by keeping his transmission rate R_t^0 in a moderate range.

All curves intersect at $R_t^0 = 15$ Mb/s. Both user 0 and 1 have the same activity of $A_s^0 = \log_2(\frac{R_t^0}{R_r^0}) = \log_2(\frac{15}{2}) = 2.91 = \log_2(\frac{7.5}{1}) = \log_2(\frac{R_t^1}{R_r^1}) = A_s^1$ at this stage. Therefore, they experience equal drop probabilities and share the bottleneck's capacity with the throughput ratio $T_0 : T_1 = R_r^0 : R_r^1 = 2 : 1$. However, appropriate values of the differentiation factor $\gamma \in \{1.5, 2, 3, 4, 5\}$ exhibit another point where a transmission rate R_t^0 lower than 15 Mb/s achieves the same fair share. This is the preferred operating point because it causes less packet loss.

Thus, ABC with an appropriate differentiation factor γ deprioritizes users who exceed their base rates by a larger factor than other users. This may effect that greedy users cannot even achieve their fair share, which is an effective defense against denial of service (DoS) attacks.

The fact that user 0 can exceed his fair share between $6\frac{2}{3}$ Mb/s $< R_t^0 <$ 15 Mb/s seems problematic at first sight. However, the reason for this phenomenon is that the activity of user 1 is larger than the one of user 0. User 1 can simply reduce his transmission rate to fully achieve his fair share. We demonstrate that in the following.

4.2.2 *Protection of Fair Shares of Lower Activity Users*

In this experiment, we study how user 1 can influence capacity sharing on the bottleneck link. He can either increase or decrease his transmission rate R_t^1. Figure 4 shows the resulting throughput values T_0 for user 0 with and without ABC. A differentiation factor of $\gamma = 3$ is configured for ABC. If user 1 increases his transmission rate to $R_t^1 = 10$ Mb/s, user 0 obtains an even larger share for $R_t^0 \in [6\frac{2}{3}; 20]$ Mb/s. If he reduces his transmission rate to $R_t^1 = 3\frac{1}{3}$ Mb/s, user 0 can hardly obtain more than $6\frac{2}{3}$ Mb/s. The very little overshoot decreases even further with increasing differentiation factor γ. Thus, ABC provides an ecosystem that allows a CBR user to achieve the fair share induced through his reference rate by sending at his fair rate. It gives incentives for users to minimize their activity in case of congestion to maximize their throughput. In Section 5 we will show that ABC leads to approximately fair bandwidth sharing with congestion-controlled TCP traffic.

We extend the analysis for 3 users with transmission rates R_t^0, R_t^1, R_t^2, and reference rates $R_r^0 = R_r^1 = R_r^2 = 1$. We choose $R_t^2 = 10$. The fair share of the users is $3\frac{1}{3}$ Mb/s. However, if user 0 transmits 4.7 Mb/s, he can achieve a higher throughput of about $T_0 = 4.06$ Mb/s independently of the transmission rate R_t^1 of user 1. This is because user 2 is deprioritized through his higher activity than user 0 and 1 so that user 0 and 1 can receive a larger share than their fair share. If user 2 lowers his transmission rate, he can increase his throughput T_2 while decreasing T_0 and T_1. Thus, basic principles also hold for multiple users.

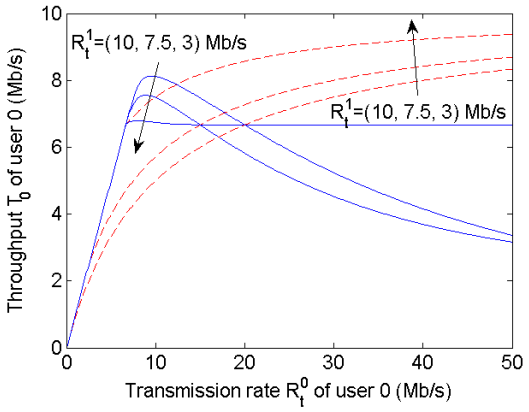

Figure 4: Throughput T_0 of user 0 on a link with $C_b = 10$ Mb/s with and without ABC. ABC parameters are $R_r^0 = 2$ Mb/s, $R_r^1 = 1$ Mb/s, and $\gamma = 3$.

4.3 Parameter Independence

We show that the base in Equation (3) may be changed and reference rates may be scaled without any effect on the capacity sharing result.

4.3.1 *Independence of Base 2*

The activity meter adds packet sizes multiplied by 2^{-A} in Equation (3) for metering. Equation (7) accounts for this fact by dividing through $\ln(2)$ when calculating the stationary activity A_s for a CBR user. This factor is contained in any activity value or average thereof. Any impact of this factor can be compensated by the choice of an appropriate differentiation factor γ in Equation (5) so that base 2 can be chosen for the meter in Equation (3) without loss of generality. If a different base was chosen, an adapted differentiation factor achieves equal bandwidth sharing results. Although the use of base 2 is not essential in Equation (3), it is important that all meters in the system use the same base for the calculation of network-wide consistent activity values.

4.3.2 *Independence of Absolute Reference Rates*

We show that the capacity sharing result depends only on the ratio of reference rates R_r^i of different users $0 \leq i < n$ with CBR traffic, but not on the absolute values of R_r^i. We assume modified reference rates $R_r^{i\prime} = x \cdot R_r^i$ for all users under the condition that $R_r^{i\prime} > R_t^i$ still holds. The stationary activity A_s^i of user i is then computed by

$$A_s^i = \log_2\left(\frac{R_t^i}{x \cdot R_r^i}\right) = \log_2(R_t^i) - \log_2(R_r^i) - \log_2(x). \quad (9)$$

The resulting average activity is

$$A_{avg} = \frac{\sum_i R_t^i \cdot (\log_2(R_t^i) - \log_2(R_r^i))}{\sum_i R_t^i} - \log_2(x). \quad (10)$$

Therefore, the difference $(A_s^i - A_{avg})$ is independent of the scalar x so that scalar multiplication of all reference rates

does neither influence drop probabilities nor throughput values in the analysis for CBR traffic.

4.3.3 Configuration of Reference Rates

We argue that reference rates should be set so small enough so that fairness can be enforced. If the sum of reference rates of potential users sharing a bottleneck link does not exceed the link's bandwidth, congestion can occur only if some users transmit faster than their reference rates. This causes packets with increased activity values that are preferentially dropped by ABC-enabled AQMs, making bandwidth sharing fairer. On the contrary, reference rates should not be set too small to avoid violation of Equation (8), again to avoid fairness degradation. However, if reference rates are not set appropriately, they do not reduce the utilization of the bottleneck link as traffic is only dropped in case of congestion.

5. RESULTS

In this section, we investigate bandwidth sharing with ABC using stochastic discrete-event simulation. We first explain the simulation methodology. Then, we validate the analytical model of Section 4.2. We demonstrate the ability of ABC to enforce per-user fairness for TCP traffic, and study its dependence on ABC configuration parameters and networking conditions. Finally, we investigate the coexistence of TCP and non-responsive traffic with ABC.

5.1 Simulation Methodology

5.1.1 Simulator and Traffic Sources

We performed simulations with INET 2.4.0 [26] in the OMNet++ network simulation framework 4.4.1 [27]. We used the Network Simulation Cradle 0.5.3 [29] 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 [22]. 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.

5.1.2 Network and AQM

We simulate the scenario depicted in Figure 5. 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 (UG0) and a user group 1 (UG1). If the experiments require a distinction between heavy and light users, the heavy users are grouped in UG0 and the light users in UG1. 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 UGi 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 $2 \cdot (D_a + D_b)$.

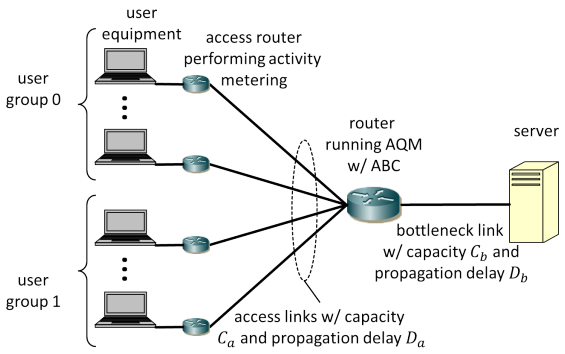

Figure 5: Simulation setup.

The bottleneck link has a simple AQM algorithm that drops or marks packets. The probability for these events depends on the current queue size Q given in packets excluding the newly arrived packet. It is determined by the following probability function:

$$p(Q) = \begin{cases} 0 & Q \le Q_0 \\ \frac{Q-Q_0}{Q_1-Q_0} \cdot p_1 & Q_0 < Q \le Q_1 \\ p_1 + \frac{Q-Q_1}{Q_2-Q_1} \cdot (1-p_1) & Q_1 < Q \le Q_2 \\ 1 & Q_2 < Q \end{cases}. \quad (11)$$

This probability function has 4 parameters: thresholds Q_0, Q_1, Q_2 and probability p_1. We denote specific functions by "pf-Q_0-Q_1-Q_2-p_1". With this 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 access links, we apply drop-tail queues also with a buffer size of 50 packets.

5.1.3 Experiment Setup and Performance Metrics

We perform multiple experiments that differ only in a few parameters. Table 1 compiles default values that are applied in all experiments if not stated differently.

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

Table 1: 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
users u_0/u_1 in UG0/1	1/10
flows f_0/f_1 per user in UG0/1	10/1
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

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 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 UGi. If $T_R = 1$ holds, the bandwidth is shared fairly among users in both groups. In case of $T_R > 1$, users in UG0 receive higher throughput than users in UG1, and in case of $T_R < 1$, users in UG1 are advantaged. Maximizing per-user fairness means bringing T_R close to 1. The widely used fairness index of Jain [14] is at most 1 and does not indicate the advantaged user group.

5.1.4 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 UDP users only, we report mean values over 100 runs with only 5 s data collection. We omit confidence intervals for the sake of better readability.

5.2 Validation of ABC Analysis for Non-Responsive Traffic

The analysis in Section 4.2 assumed an AQM probability p that is constant for all packets which in fact changes over time. We validate the analysis by providing simulation results for selected data points in Figure 3. We simulate two users, one with reference rate $R_r^0 = 2$ Mb/s and the other with $R_r^1 = 1$ Mb/s. Each of them has a single CBR flow sending UDP traffic with rates $R_t^0 \in \{10, 20\}$ Mb/s and $R_t^1 = 7.5$ Mb/s. We perform experiments with several differentiation factors $\gamma \in \{0, 0.5, 1, 1.5, 2, 3, 4, 5\}$. The results are presented as dots in Figure 3. They match the analytical curves quite accurately, confirming the insights and conclusions drawn in Section 4.

5.3 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 also heterogeneous capacity sharing and can decrease upload times significantly. In all conducted experiments, the bandwidth utilization of the bottleneck link is 100%. Thus, ABC shows no negative impact on link utilization.

5.3.1 Impact of ABC Parameters

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 2 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 adverse conditions ($u_0/u_1 = 1/10$ and $D_b = 50$ ms) a relatively large bandwidth share for light users ($T_R = 1.37$). In this case, the light users achieve lower throughput than the heavy user because they cannot fully exploit their relatively large 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 have a better chance to 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 ($T_R = 0.89$ for $u_0/u_1 = 9/90$ and $D_b = 5$ ms, $T_R = 0.86$ for $u_0/u_1 = 9/90$ and $D_b = 50$ ms). This is not ideal but we prefer it to throughput ratios larger than 1 because heavy users could lower their transmission rate to improve their 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 2: Throughput ratio T_R depending on differentiation factor γ.

D_b	users	differentiation parameter γ					
(ms)	u_0/u_1	0	1	2	3	4	5
5	1/10	9.80	1.85	1.24	1.04	0.95	0.90
	9/90	10.26	1.35	0.99	0.89	0.83	0.81
50	1/10	10.04	2.74	1.71	1.37	1.21	1.11
	9/90	10.40	1.53	0.99	0.86	0.78	0.72

Impact of Activity Inertia I.

We study the impact of the activity inertia I for $\gamma = 3$ and various congestion levels. Table 3 shows that increasing inertia I decreases the throughput ratio T_R for all congestion levels. However, increasing inertia I also increases the adaptation time of the activity meter to reduced load conditions (see Section 4.1). Therefore, we take $I = 0.6$ s as compromise in our simulations. It allows to use small reference rates of 0.02 Mb/s without the risk of oscillating activity values.

Table 3: Throughput ratio T_R depending on inertia I.

D_b (ms)	u_0/u_1	activity inertia I (s)					
		0.15	0.3	0.6	1.2	2.4	4.8
5	1/10	1.12	1.06	1.04	1.02	1.01	1.00
	9/90	1.01	0.94	0.89	0.86	0.84	0.81
50	1/10	1.43	1.38	1.37	1.33	1.30	1.25
	9/90	1.17	0.88	0.86	0.84	0.82	0.81

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 6 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. They are 0.1 Mb/s for 9/90 users and 1 Mb/s for 1/11 users, and they are independent of the bottleneck delay D_b. These values confirm that the sum of reference rates of all users should not exceed the bottleneck's bandwidth. However, the 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, only an upper bound on the active number of users on a bottleneck resource is needed for the configuration of reference rates. This confirms our findings for non-responsive traffic in Section 4.3.2.

Impact of Activity Weight w_A.

The AQM computes the average activity A_{avg} from activity values in received packets using an EWMA. Its activity weight w_A determines how quickly A_{avg} adapts to changed activity values in terms of received packets. Thus, the adaptation speed depends on w_A and the packet arrival rate. 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.

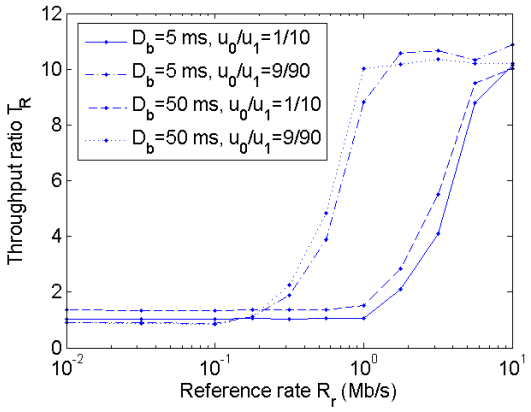

Figure 6: Throughput ratio T_R depending on reference rate R_r.

Impact of Granularity g.

Activity values can be coded into packet headers only with limited accuracy. They may be expressed as integral multiples of a basic granularity g. We round up activity values to $A_g = \lceil \frac{A}{g} \rceil \cdot g$. Thus, reported activity values are at most g larger than the exact value, in no case they are smaller. As a consequence, slightly different transmission rates ($R_r \cdot 2^{n\dot{g}} - \varepsilon$ and $R_r \cdot 2^{n\cdot g} + \varepsilon$) yield rounded activity values that differ by g, indicating that the rates differ by a factor of 2^g. Conversely, transmission rates that differ by a factor of almost 2^g may yield the same signalled activity ($R_r \cdot 2^{n\cdot g} + \varepsilon$ and $R_r \cdot 2^{(n+1)\cdot g} - \varepsilon$). Since ABC cannot produce better throughput ratios than 2^g for CBR traffic under adverse conditions, g should be chosen low enough. Additional experiments have shown that the granularity has less impact on bandwidth sharing with TCP traffic because rates of heavy and light users adapt. A granularity of $g = 1$ had only a minor impact on throughput ratios in our experiments.

5.3.2 *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 4 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-10-23-24-0 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 probability 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-10-{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.

Table 4: Impact of probability functions.

D_b / u_0/u_1	5 ms 9/90		50 ms 1/10	
prob. function	T_R	Q_{avg} (pkts)	T_R	Q_{avg} (pkts)
pf-10-23-24-0	10.98	22.09	9.75	15.44
pf-10-22-24-0	1.59	21.54	4.27	15.97
pf-10-17-24-0	0.93	18.89	2.32	13.42
pf-10-11-24-0	0.89	15.78	1.82	8.32
pf-11-17-24-0.1	0.90	18.46	1.38	9.83
pf-11-17-24-0.01	0.89	18.84	1.37	11.63

Table 5: Impact of ECN-Enabled Flows.

scenario		$D_b = 5$ ms		$D_b = 50$ ms	
ECN	u_0/u_1	T_R	Q_{avg}	T_R	Q_{avg}
off	1/10	1.04	16.08	1.37	12.14
	3/30	0.90	17.54	1.02	15.96
	9/90	0.89	18.96	0.86	18.15
on $Q_{max} =$ 24 pkts	1/10	1.04	17.48	2.07	7.09
	3/30	0.95	21.70	2.14	14.23
	9/90	2.68	22.03	1.30	21.63
on $Q_{max} =$ 50 pkts	1/10	1.97	14.41	2.30	7.06
	3/30	1.04	31.79	5.00	14.13
	9/90	10.33	48.81	5.49	41.83

Function pf-10-22-24-0 allows differentiation only for occupancy 23, which already reduces the throughput ratio from 9.75 to 4.27 compared with the step function. Functions pf-10-17-24-0 and pf-10-11-24-0 provide 6 and 12 occupancy levels for differentiation and reduce the throughput ratio further to 2.32 and 1.82, respectively. We observe for 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 even 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 value in case of low-congestion. Therefore, we consider double-slope functions (pf-11-17-24-{0.01,0.1} whose probability first increases linearly 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 $T_R = 1.37$. We chose pf-11-17-24-0.01 as default for our experiments. More complex functions are imaginable.

The high-congestion scenario 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 contributes to lower throughput ratios. The step function is again an exception as it disables ABC.

5.3.3 Impact of ECN-Enabled Flows

We study ABC for ECN-enabled 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 even dropped without any possibility for differentiation. Throughput ratio and average queue lengths are compiled for this setting in Table 5 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, also heavy users fill up high-load positions of the queue in case of congestion if flows do not reduce their rates quickly enough. Then, packets from heavy and light users see the same tail-drop probability. This diminishes the degree of differentiated treatment for heavy and light users and increases the throughput ratio. As another consequence, the average queue length Q_{avg} may increase, especially in high-load scenarios. In case of very low congestion, i.e., long D_b and only a few users, the throughput ratio also increases because marking packets instead of dropping them increases in particular the throughput for heavy users.

With a buffer size of $Q_{max} = 50$ packets, the throughput ratio increases even further, especially for high congestion. 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. In the worst case, this disables ABC yielding throughput values around 10. 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 packets should be dropped instead of being marked.

5.3.4 Heterogeneous Capacity Shares

ABC is designed to support heterogeneous capacity shares by configuration of appropriate reference rates. We investigate to what extent this bandwidth share can be exploited by a privileged user with a single TCP flow. We assume UG0 holds only the privileged user with a single flow and a scaled reference rate of $R_r^0 = s \cdot R_r^1$. UG1 holds multiple users with a single flow each. Table 6 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 bandwidth share when using multiple flows.

5.3.5 Reduced Upload Times

ABC can reduce upload times for occasional transactions of moderate size. We model this by a single "probe" user

Table 6: 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
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
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

who transmits a probe of 1 MB on application layer. Background traffic is produced by u_1 users sending with f_1 saturated TCP flows each. Upload times for the probe are compiled in Table 7. 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 7: Upload times (s) with and without ABC.

D_b (ms)	ABC	B (MB)	$f_1 = 1$		$f_1 = 4$	
			$u_1 = 10$	$u_1 = 20$	$u_1 = 10$	$u_1 = 20$
5	no	n/a	10.3	20.2	34.2	61.4
	yes	0	9.1	19.7	11.5	18.8
		0.25	8.7	15.7	9.6	15.7
		0.5	6.5	11.3	7.5	11.6
		0.75	5.1	7.7	5.5	7.9
		1	2.8	3.0	2.9	3.2
		1.25	2.7	2.7	2.7	2.7
50	no	n/a	20.3	29.2	47.7	85.2
	yes	0	9.5	18.5	12.2	21.7
		0.25	7.2	14.8	10.6	18.2
		0.5	5.9	11.4	8.6	14.5
		0.75	3.4	7.3	6.6	10.6
		1	2.8	3.7	4.3	6.4
		1.25	2.6	3.3	4.0	5.3

We perform the same experiments with ABC while varying the burst allowance for the probe user. Table 7 shows only a slight reduction of the upload time for background users with $f_1 = 1$ flow and burst allowance $B = 0$ bytes. If background users have $f_1 = 4$ flows, the upload time is already substantially reduced because ABC partitions the bottleneck bandwidth equally among users. 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.

5.4 Performance of ABC with TCP and Non-Responsive Traffic

We consider a single heavy user sending CBR UDP traffic at rate R_t^0, competing with $u_1 = 10$ light users with a single TCP flow each. Without ABC, TCP traffic roughly consumes the capacity that is left over by UDP traffic, leading to low throughput for TCP users. Figure 7 shows that this is different with ABC. For $\gamma = 3$, the throughput ratio T_R exhibits an overshoot around the fair share, but quickly decreases below 1 for increasing R_t^0, i.e., TCP users obtain a larger capacity share than UDP users with too large UDP transmission rates R_t^0. The throughput ratio increases again for very large R_t^0. Further experiments have shown that the latter effect can be avoided by a triple-slope probability function with the third slope starting at a high probability value and ending at 1. With this modification, ABC well protects TCP traffic against non-responsive sources sending at a rate larger than their fair share. Thereby ABC may also mitigate denial of service attacks.

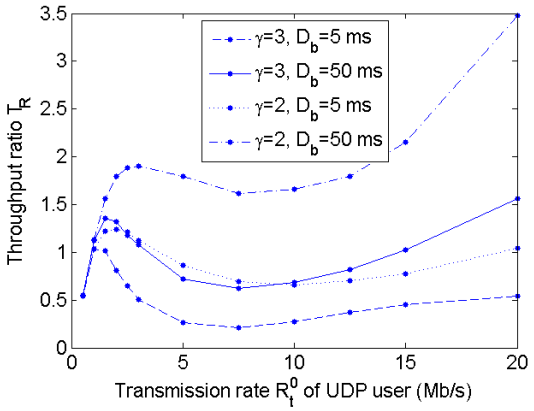

Figure 7: Throughput ratio T_R of a UDP user and 10 TCP users.

A differentiation factor of $\gamma = 2$ causes larger throughput ratios, allowing again UDP users to monopolize bandwidth under certain conditions. Also the improved AQM probability function cannot prevent that the UDP user gets 50% more bandwidth than TCP users for $R_t^0 = 20$ Mb/s and $D_b = 50$ ms. This confirms our recommendation for $\gamma = 3$.

6. CONCLUSION

We presented activity-based congestion management (ABC) which enables users to obtain a fair share of a bottleneck capacity in case of congestion, i.e., a share that is approximately proportional to their configured reference rates. ABC consists of an activity meter equipping the user's traffic with activity information, and an extension of active queue management (AQM) in routers or switches that leverages a packet's activity to adapt the AQM's loss/mark probability. ABC is scalable because switches do not require per-user states or additional queues.

We proposed an algorithm for ABC's activity meter which is based on a token-bucket. It determines the logarithmic value of the factor by which a user's transmission rate exceeds his reference rate. This algorithm may also be used for fast rate measurement in other contexts.

We studied the behavior of the activity meter and of ABC in the presence of non-responsive traffic by mathematical analysis. ABC gives significantly better treatment to flows being closer to their fair share, incentivizing the use of congestion control algorithms. We validated the analysis by simulation and considered ABC with TCP traffic. We showed that heavy and light users can obtain with ABC about equal bandwidth shares on a congested link and investigated configuration and stability issues for multiple networking scenarios. ABC supports heterogeneous capacity shares, reduces upload times for moderate transactions significantly, and protects TCP traffic against overload of non-responsive traffic.

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.

7. REFERENCES

[1] R. Adams. Active Queue Management: A Survey. *IEEE Comm. Surveys & Tutorials*, 15(3), 2013.

[2] S. Baillargeon and I. Johansson. ConEx Lite for Mobile Networks. In *ACM CSWS*, 2014.

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

[4] B. Briscoe. Flow Rate Fairness: Dismantling a Religion. *ACM CCR*, 37(2), Apr. 2007.

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

[6] B. Briscoe. Initial Congestion Exposure (ConEx) Deployment Examples. `http://tools.ietf.org/html/draft-briscoe-conex-initial-deploy`, Oct. 2011.

[7] B. Briscoe et al. Policing Congestion Response in an Internetwork using Re-feedback. In *ACM SIGCOMM*, 2005.

[8] 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.

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

[10] Broadband Internet Technical Advisory Group (BITAG). Real-Time Network Management of Internet Congestion. Technical report, Oct. 2013.

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

[12] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4), Aug. 1993.

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

[14] R. Jain et al. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. Technical Report TR-301, DEC, Sept. 1984.

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

[16] D. Kutscher et al. Mobile Communication Congestion Exposure Scenario. `http://tools.ietf.org/html/draft-ietf-conex-mobile`, Sept. 2014.

[17] M. Mathis and B. Briscoe. Congestion Exposure (ConEx) Concepts, Abstract Mechanism and Requirements. `http://tools.ietf.org/html/draft-ietf-conex-abstract-mech`, Oct. 2014.

[18] K. Nichols and V. Jacobson. Controlling Queue Delay. *ACM Queue*, 10(5), May 2012.

[19] R. Pan et al. CHOKEs: a Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *IEEE Infocom*, 2000.

[20] R. Pan et al. PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem. In *IEEE HPSR*, 2013.

[21] B. Raghavan et al. Cloud Control with Distributed Rate Limiting. In *ACM SIGCOMM*, Aug. 2007.

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

[23] Sandvine Incorporated ULC. Network Congestion Management: Considerations and Techniques – An Industry Whitepaper. Technical report, Oct. 2015.

[24] S. Shalunov et al. RFC6817: Low Extra Delay Background Transport (LEDBAT), Dec. 2012.

[25] A. Shieh et al. Sharing the Data Center Network. In *USENIX NSDI*, 2011.

[26] A. Varga. INET-2.4.0 released. `http://inet.omnetpp.org/`, June 2014.

[27] A. Varga. OMNeT++ 4.4.1 released. `http://www.omnetpp.org/`, Oct. 2014.

[28] D. Wagner. Congestion Policing Queues – A New Approach to Managing Bandwidth Sharing at Bottlenecks. In *CNSM*, 2014.

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