Active Queue Management Based on Congestion Policing (CP-AQM)

Michael Menth and Sebastian Veith

University of Tuebingen, Chair of Communication Networks,
Sand 13, 72076 Tuebingen, Germany
{menth@,sebastian.veith@}uni-tuebingen.de

Abstract. Buffers in switches or routers are used to achieve sufficiently high utilization of transmission resources but permanently filled buffers add excessive queuing delay to communication. Active queue management (AQM) accommodates infrequent traffic bursts but avoids a standing queue. Currently, many new AQM mechanisms are discussed in the IETF. In this work, we propose a new AQM mechanism based on the idea of congestion policing. We evaluate its performance for various networking scenarios and transport protocols, and illustrate the impact of its parameters.

Keywords: Bufferbloat, active queue management, congestion policing

1 Introduction

Bufferbloat [14] is the phenomenon that end-to-end delay in the Internet can be very high due to large buffers in switching nodes on the paths. The delay occurs if large buffers get filled as this increases the queueing delay of packets. This observation has fueled the discussion of new active queue management (AQM) mechanisms for the Internet. Random Early Detection (RED) has been proposed as AQM for the Internet [6] already in 1998, but so far there is only little deployment which is also due to its rather difficult configuration. Therefore, the IETF has started a new working group on "Active Queue Management and Packet Scheduling" (AQM) [18] where novel AQM algorithms are investigated and standardized.

Congestion policing (CP) is the idea that traffic gets policed if a certain amount of congestion is exceeded. The idea was first introduced in [10] to improve the fairness among competing users on a remote link under congestion conditions. Packets of a user get dropped when he causes more CE-marked packets than his congestion allowance in form of a configured rate plus some tolerance. A

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

quantitative evaluation was missing. In this work, we take a step back and apply this principle to the traffic entering a local queue: if more congestion occurs than a predefined amount, some packets are dropped. We propose an appropriate definition of congestion and design a congestion policer. The congestion policer is based on a token bucket and controls all traffic entering the queue. This yields the CP-AQM mechanism. We propose suitable token bucket parameters and investigate CP-AQM by means of simulation in different networking scenarios, for different transport protocols, and for different traffic types on a 10 Mb/s link. We recommend congestion parameters for which low average queue lengths and high link utilization are observed in all considered experiments.

In Section 2, we revisit related work and in Section 3 we present the design of CP-AQM. In Section 4 we explain our simulation setup and methodology and present comprehensive results that give insight into the performance of CP-AQM and the impact of its parameters. Section 5 summarizes the results and gives an outlook on future work.

2 Related Work

Buffers are deployed in almost all devices of the Internet as they are needed for efficient packet multiplexing. Some of them are oversized which may cause persistent excessive delay if a standing queue occurs. This phenomenon is called bufferbloat [14]. It is due to the fact that TCP cannot recognize an increasing queue unless there is packet loss or significant delay. While some authors are rather doubtful about the prevalence of bufferbloat and its impact [2, 17], bufferbloat has been demonstrated in cellular networks [21]. The authors of [9] pointed out many sources contributing to Internet latency and countermeasures.

AQM mechanisms generally reduce queue lengths and fight against bufferbloat. Random Early Detection (RED) [13] was among the first AQM mechanisms for the Internet and has been proposed as a standard [6]. The survey in [1] nicely categorizes a multitude of other AQMs, many of them using the same principle as RED. The "Adaptive Virtual Queue (AVQ) is a rate-based AQM that attempts to maintain input (arrival) rate at a desired utilization." [1]. "Stabilized Virtual Buffer (SVB) uses both packet arrival rate and queue-length as part of its congestion indicator and attempts to keep the the packet arrival rate and the queue-length around their individual target values." [1]. These mechanisms are most similar to the proposed CP-AQM scheme, but differ in a significant detail: their virtual queue monitors traffic instead of congestion.

Since low delay has become more important in the recent years and since RED has not been vastly deployed, the IETF working group AQM [18] has been established. Its objective is to produce new recommendations for AQM in the Internet [4]. Three novel AQM mechanisms have been presented in that course: CoDel, PIE, and GSP. CoDel [27, 28] stands for "Controlled Delay". It is mostly considered in combination with stochastic fair queuing (SFQ) to isolate flows against each other [16]. However, this combination is basically applicable to any AQM mechanism as buffer management and scheduling are orthogonal

to each other [5]. One advantage of CoDel over RED is that it copes with variable capacity links which is relevant in wireless networks. The authors of [23] provide a comparison between CoDel and RED. Interactions between CoDel and LEDBAT, a transmission protocol for background traffic, have been studied in [15]. The authors of [32] have presented a software-defined implementation in an FPGA for RED and CoDel to support 10 Gb/s. PIE [29, 30] stands for "Proportional Integral controller Enhanced". PIE was designed to yield more efficient implementations than CoDel as it does not require timestamps. It has already been tested for DOCSIS systems [12, 36, 37]. A comparison between PIE and CoDel for DOCSIS is provided in [26]. Another comparison between PIE, CoDel, and ARED is presented in [22]. GSP is short for "Global Synchronization Protection for Packet Queues" [25]. The basic operation of GSP relies on fixed bandwidth. There is an adaptation of GSP for scenarios with higher loads and for queues with variable capacity. WQM [31] is a novel queue management system designed for IEEE 802.11n networks to fight against bufferbloat resulting from variable server rates.

CP-AQM leverages CP. CP limits the maximum congestion a user or traffic aggregate can cause by packet drops that are enforced by a policer. The idea of CP was first introduced in [10] and later in [20] and [8]. As it was originally intended for distributed systems, information about congestion caused by a flow and observed by the receiver is returned back to the sender that inserts information about this observed congestion into the IP header. This information is known as re-feedback [10] or congestion exposure (ConEx). An IETF working group [19] was established to standardize this protocol as experimental standard. ConEx information is intended to perform congestion management through CP [11]. Use cases are data centers [7] or backhaul networks for mobile access networks [24]. A modified version of ConEx-based CP has been presented in [3]. So far, there is no local application of the CP principle and there is no quantitative evaluation of CP. This work suggests such a local application and investigates its performance by means of packet-based simulation.

3 Design of CP-AQM

In this section we introduce CP-AQM. We first give an appropriate definition of congestion, then explain the design and operation of CP-AQM, and eventually derive configuration parameters.

3.1 Congestion Function

Congestion is a rather informal term denoting some form of overload on a link. However, we need a quantitative definition to measure it. Briscoe quantified it as a rate of lost and CE-marked packets on a link [8]. As packet loss occurs only under extreme load, and CE-marking of packets depends on the configuration of the marking algorithm, this definition is not appropriate for our purpose. Instead

we provide the following congestion function that depends on the current queue occupancy x in bytes:

$$c(x) = \begin{cases} 0 & x < T_c \\ 1 + \frac{x - T_c}{Q_{max} - T_c} \cdot (c_{max} - 1) & x \geq T_c. \end{cases} \quad (1)$$

Thereby Q_{max} is the capacity of the queue in bytes and T_c is the congestion threshold: queue sizes below that threshold are considered uncongested and queue sizes equal or larger are considered congested. Congestion starts with $c(T_c) = 1$, the severity of congestion linearly increases with the queue length, and reaches c_{max} for a full queue. Therefore, c_{max} should be larger than 1.

3.2 Congestion Policer

The congestion policer drops packets if the congestion on the link exceeds a configured congestion allowance or if the packet does not fit into the queue. The congestion allowance consists of token bucket parameters: rate R_{CA} (bit/s) and bucket size B_{CA} (bytes). That means, the policer has a bucket of size B_{CA} that is continuously refilled with tokens at a rate of R_{CA}. Let B be the size of a packet on the link including all overheads. If a packet arrives at the queue and the queue currently holds x bytes, then the packet contributes $B \cdot c(x)$ congestion (bytes). If the fill state of the bucket is at least $B \cdot c(x)$, the packet is accepted for sending and the fill state of the queue is reduced by $B \cdot c(x)$; otherwise, the packet is dropped by the policer without changing the bucket fill state. If the packet must be dropped because the queue is full, the fill state is not reduced.

There is an important difference between CP-AQM and conventional token bucket (or virtual queue) based policers. Conventional token bucket policers meter traffic and drop packets if the traffic stream exceeds a configured rate by some configured tolerance. CP-AQM meters congestion and drops packets if the congestion stream exceeds the configured congestion allowance. Congestion exists only if the fill state of the queue is sufficiently high, but then the congestion rate of a traffic stream may exceed its traffic rate.

3.3 Parametrization

Obvious configuration parameters of CP-AQM are the congestion threshold T_c, the maximum congestion c_{max}, and the congestion allowance parameters rate R_{CA} and bucket size B_{CA}. Let C be the link bandwidth.

If the queue is full and traffic is sent at link speed C, then a maximum congestion rate of $C \cdot c_{max}$ can be generated. Thus, for the policer to be effective, the congestion allowance rate R_{CA} must be smaller than that value. To enable the policer to avoid permanent queue occupancies of size T_c or larger, the congestion allowance rate R_{CA} should be at most C. If the bucket becomes empty, the queue is above the congestion threshold T_c. If the sender sends with at least link bandwidth C in this situation and its congestion allowance rate R_{CA} is set to a

value smaller than C, then the policer drops at least a fraction of $\frac{C-R_{CA}}{C}$ packets. To make the loss rate in that situation only dependent on the arrival rate and the fill state of the queue, but independent of R_{CA}, we choose $R_{CA} = C$.

The policer should allow the traffic to fill the entire queue, but only for short time. In particular, filling the queue with a single burst should be possible without causing the policer to drop any packets. The resulting amount of congestion can be approximated by $(Q_{max} - T_c) \cdot \frac{1+c_{max}}{2}$, which is a reasonable lower bound on the bucket size. This value is maximized for $T_c = 0$ and $c_{max} = 2$ in our experiments, which leads to $\frac{3}{2} \cdot Q_{max}$. If the physical queue has room for $Q_{max} = 45$ KB and IP packets are 1.5 KB large, a bucket size of $B_{CA} = 45 \text{ KB } \cdot \frac{1.507\text{KB/pkt}}{1.5 \text{ KB/pkt}} \cdot 1.5 = 67.815$ KB is needed after taking the PPP overhead in our simulation into account which is also respected for the congestion calculation. In the special case of $T_c = 0$ the bucket needs to be even one packet larger because then all packets cause congestion, even in the presence of an empty queue. We use this rule to configure B_{CA} for all experiments because additional runs showed that this bucket size is sufficient to produce high utilization and that larger bucket sizes cannot increase the utilization significantly. Thus, from a certain bucket size on, CP-AQM is rather insensitive to the congestion allowance bucket size B_{CA}. After all, the congestion threshold T_c and the maximum congestion c_{max} remain as configuration parameters for CP-AQM.

4 Queueing Behavior with CP-AQM

We investigate the impact of the configuration parameters of CP-AQM on average queue length and utilization on a 10 Mb/s link. To that end, we consider various networking scenarios, transport protocols, and traffic types. We first describe the simulation methodology and experiment setup. We illustrate the queuing behavior of both TCP New Reno and TCP Cubic connections in various networking conditions using tail-drop buffer management as baseline. Then, we show how CP-AQM performs under various conditions for non-reactive traffic generating persistent severe congestion. Eventually, we investigate the impact of CP-AQM's configuration parameters for different networking scenarios and give recommendations.

4.1 Simulation Methodology and Setup

We used the INET framework [33] of OMNeT++ [34] for simulations. As we do not trust INET's TCP implementation, we use the Network Simulation Cradle [35] based on which INET allows to integrate Linux networking stacks including TCP New Reno and Cubic.

We briefly describe the simulation setup. Users are connected to a server via a private, fast access link and a shared, slow bottleneck link. The access links have capacity $C_a = 1$ Gb/s and one-way propagation delay $D_a = 0.1$ ms while the shared bottleneck link has capacity of $C_b = 10$ Mb/s and a one-way propagation delay D_b. The access links do not cause any packet loss. The queue length on

the bottleneck link is $Q_{max} = 45$ KB which corresponds to 30 maximum-size IP packets.

For non-reactive traffic with a given average rate, we consider periodic traffic and traffic with exponential inter-arrival times. TCP performance significantly depends on round-trip time (RTT) and packet loss. Especially the latter strongly depends on the number of flows on the bottleneck link. Therefore, we look at different networking scenarios with $n \in \{1, 16\}$ TCP flows and $D_b = \in \{5, 50\}$ ms. These values cause round trip times of at least 10.1 ms or 100.1 ms propagation delay plus transmission and queueing delay which may be significant.

Our simulation features a bottleneck link with $C_b = 10$ Mb/s over which PPP frames are transmitted. Thus, packets come with 7 bytes overhead for PPP header, 20 bytes overhead for IP header, and 8 or 20 bytes overhead for UDP or TCP header. The maximum transfer unit for an IP packet is 1500 bytes. UDP traffic is constant bit rate with 1472 bytes UDP payload per packet.

Each data point in the figure is an average gained from at least 100 simulation runs, each of them pertains to a simulation time of at least 100 s with a preceding 10 s warmup phase. Flows were randomly started within the first 5 s of the simulation. In the case of a single flow, we conducted at least 1000 simulation runs with a duration of at least 1000 s.

4.2 Tail-Drop Buffer Management

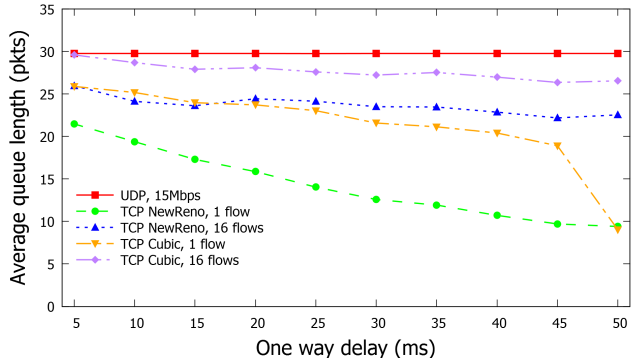
We illustrate the impact of tail-drop buffer management on the queueing behavior to provide a baseline for CP-AQM. We perform experiment series for different traffic types. Figures 1(a) and 1(b) show the average queue length and the utilization of the bottleneck link for different one-way delays.

We first consider constant-bit rate UDP traffic with 15 Mb/s on PPP layer. As the offered traffic is significantly larger than the available bandwidth, the link utilization is 100% and the queue is always fully occupied regardless of the one-way delay.

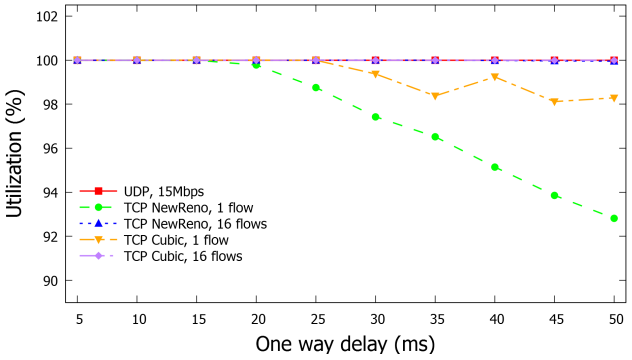
A single TCP New Reno connection also achieves full utilization, but only up to a one-way delay of about $D_b = 20$ ms while for $D_b = 50$ ms the link utilization degrades to 92.7%. TCP Cubic is more aggressive and fills the pipe up to $D_b = 25$ ms and reaches a utilization of about 98% even for $D_b = 50$ ms. With increasing one-way delay, the average queue length decreases from 20 packets to 10 packets for a single TCP New Reno connection. TCP Cubic leads to larger average queue length than TCP New Reno as its congestion control algorithm increases its sending rate more quickly after packet loss.

More TCP connections lead to more congestion. As a result, the link capacity can be fully used by 16 TCP flows even for a $D_b = 50$ ms, regardless of the TCP variant. The average queue length increases to values between 26 and 30 packets for TCP Cubic and 22.5 and 25.5 packets for TCP New Reno. Also packet loss becomes significant under these conditions and varies between 4% and 12% for TCP Cubic and between 2.5% and 9% for TCP New Reno (not shown in the figures).

The fact that TCP can lead to a full queue over long time, which is the case for average queue lengths above 15 packets, can be considered as bufferbloat. In particular for $D_b = 5$ ms, TCP Cubic keeps the buffer almost constantly full which is not necessary for efficient packet multiplexing, but adds delay which is especially annoying when additional real-time traffic is also carried over the bottleneck link.



(a) Average queue length on the bottleneck link.



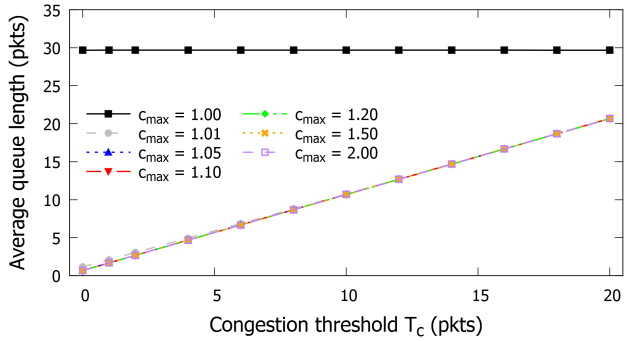
(b) Utilization on the bottleneck link.

Fig. 1. Impact of networking parameters on the performance of TCP traffic with tail-drop buffer management.

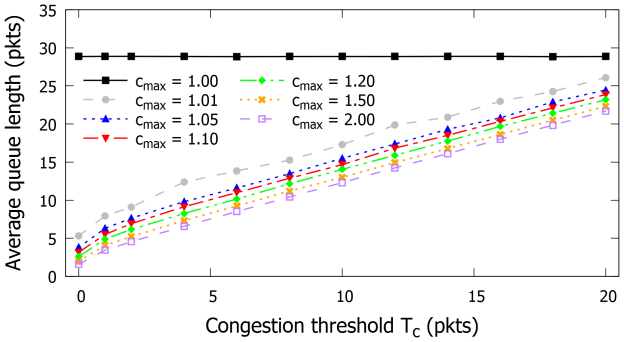
4.3 Non-Responsive Traffic with CP-AQM

We study how configuration parameters of CP-AQM influence the queueing behavior of non-responsive traffic that causes significant overload. The link utilization was 100% for all experiment series and all investigated configuration parameters.

Figure 2(a) illustrates the average queue length for 15 Mb/s constant bit rate UDP traffic on PPP layer with constant packet inter-arrival times. For $c_{max} > 1$ the average queue length increases linearly with the congestion threshold T_c but the exact value of c_{max} has no impact. We analyze the system behavior. Since the traffic rate exceeds the link bandwidth, the queue initially increases so that the congestion function yields values larger than 1. As a result, the bucket is drained faster than it is refilled so that it eventually holds too few tokens to accept a packet. From this point on, less traffic than link bandwidth can be accepted due to $c(x) > 1$ for $x > T_c$ so that the physical queue shrinks to T_c for which $c(T_c) = 1$ holds. At this stage, a traffic rate of exactly C_b can be accepted so that excess traffic is dropped. If the queue size falls below T_c due to a packet drop, the fill state of the bucket slightly increases, which allows a packet train of consecutive packets to be accepted so that the queue exceeds T_c by a very few packets. Thus, the queue length oscillates with a very low amplitude around T_c.



(a) Constant packet inter-arrival times.



(b) Exponential packet inter-arrival times.

Fig. 2. Impact of CP-AQM configuration parameters on the average queue length on the bottleneck link for non-responsive UDP traffic with a rate of 15 Mb/s.

Figure 2(b) shows that average queue lengths for non-responsive traffic with exponential packet inter-arrival times are larger than for periodic traffic. This is due to the fact that the bucket can refill significantly during inter-arrival times that are larger than average, which allows the queue to grow larger afterwards. We provide a simple model for this phenomenon. We assume that the queue size falls below T_c for t_{below} time which increases the bucket size by $C_b \cdot t_{below}$ tokens. If the queue size above T_c is on average Q^{avg}_{above}, the duration t_{above} of the queue size above T_c can be calculated by

$$t_{above} = \frac{C_b \cdot t_{below}}{C_b \cdot \frac{Q^{avg}_{above} - T_c}{Q_{max} - T_c} \cdot (c_{max} - 1)}. \tag{2}$$

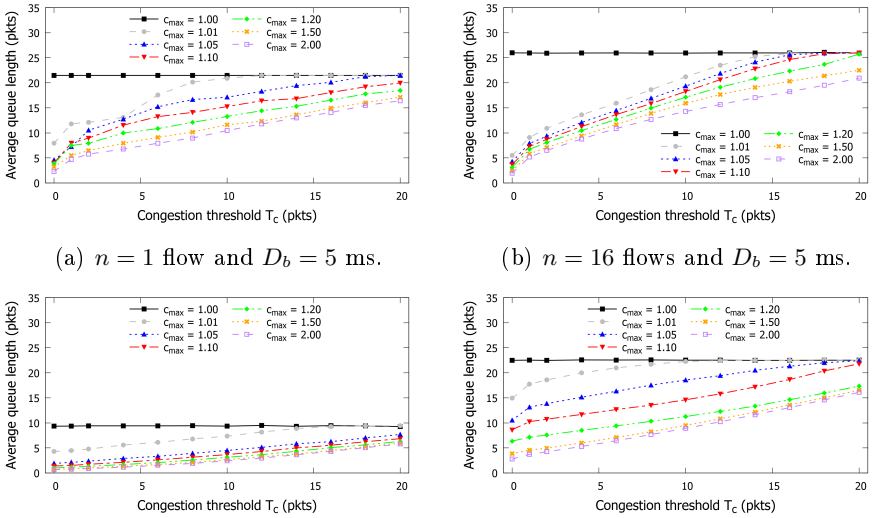
Example values $t_{below} = 2.4$ ms, $c_{max} = 1.2$, $T_c = 10$ and $Q^{avg}_{above} = 15$ yield a bucket increase by 3000 bytes (2 packets) after which the queue size can stay $t_{above} = 48$ ms at $Q^{avg}_{above} = 15$ packets on average. Thus, a very short time (2.4 ms) of the queue size below T_c can allow the queue to stay for long time (48 ms) above T_c. If we assume an average queue length of 9 packets during t_{below}, this leads to an overall average queue length of 14.7 packets.

The observed deviations increase with smaller maximum congestion c_{max}. Thus, with exponential packet inter-arrival times the queue length oscillates more strongly around T_c, and c_{max} influences the size of the amplitudes. Experiments with 12 Mb/s lead to larger average queue sizes because this makes larger packet inter-arrival times more likely. Conversely, 20 Mb/s lead to smaller average queue sizes.

For a maximum congestion of $c_{max} = 1$ we observe in both figures a straight line on the level of the average queue length obtained for tail-drop. This parameter value does not cause the policer to drop any packets so that the queuing behavior is independent of the congestion threshold T_c and equal to the one for drop-tail. This parameter value effects that the congestion contributed by a packet is exactly its size on the channel (PPP frame size in our simulation). As the congestion allowance rate R_{CA} equals the link bandwidth C_b, the number of tokens missing in the token bucket of the policer can be at most the current queue length (minus the congestion threshold T_c, plus the PPP header overhead of the stored IP packets, to be accurate). As we have chosen the token bucket size sufficiently larger than the queue size, the token bucket cannot run empty so that the policer cannot drop packets. In the following, we use the curve for $c_{max} = 1$ as reference for tail-drop.

4.4 TCP Traffic with CP-AQM

An AQM should be configured such that it performs well for all relevant traffic patterns. As we observed in the preceding section a significant impact of the number of TCP flows, the one-way delay, and the TCP version on average queue length, we investigate the impact of CP-AQM's configuration parameters on queuing behavior for 8 combinations consisting of $n \in \{1, 16\}$ flows, $D_b \in \{5, 50\}$ ms, and TCP version $\in \{\text{New Reno}, \text{Cubic}\}$.



(a) $n = 1$ flow and $D_b = 5$ ms.

(b) $n = 16$ flows and $D_b = 5$ ms.

(c) $n = 1$ flow and $D_b = 50$ ms.

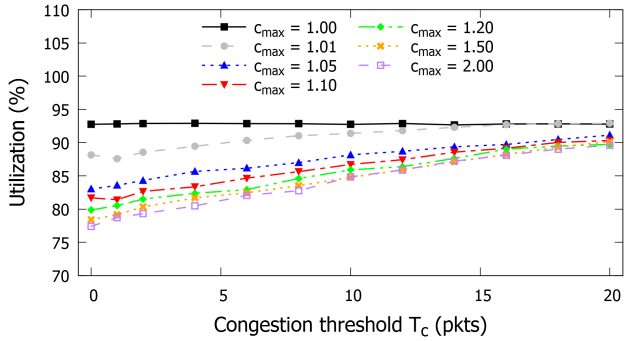
(d) $n = 16$ flows and $D_b = 50$ ms.

Fig. 3. Average queue lengths on the bottleneck link for TCP New Reno.

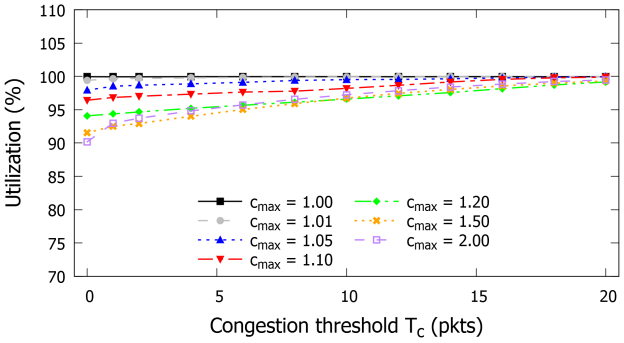
Experiments with TCP New Reno Figure 3(a) shows the average queue length on the bottleneck link for $n = 1$ TCP New Reno connection and $D_b = 5$ ms. It clearly increases with an increasing congestion threshold T_c and with a decreasing maximum congestion c_{max}. We briefly discuss these findings.

The fact that the average queue length increases with the congestion threshold is rather intuitive. A packet contributes to congestion only if the queue occupation is at least the congestion threshold T_c at its arrival. Only then the packet can be policed; otherwise it does not contribute any congestion. Thus, for larger values of T_c, the queue can grow larger without consuming tokens from the bucket, which effects policer drops only at larger queue sizes. The average queue length is shorter for larger maximum congestions c_{max} than for small ones. This is because large values of c_{max} generate a similar congestion rate already at lower queue sizes compared to small values of c_{max} so that CP starts dropping at lower queue sizes. The link utilization reaches mostly 100% except for $T_c \in \{0, 1\}$ packets and large c_{max} values where utilization is between 86% and 99% (not shown by figures). Thus, CP-AQM can limit the average queue length for $D_b = 5$ ms without sacrificing hardly any utilization.

Figures 3(b)–3(d) show the results for more TCP New Reno connections, or longer one-way delays, or both. They are qualitatively the same, but differ in detail. For short one-way delays $D_b = 5$ ms or for only a single $n = 1$ TCP flow, the contention for queue space is low enough so that a low congestion threshold $T_c \leq 5$ can enforce small average queue lengths. However, for $D_b = 50$ ms and



(a) $n = 1$ TCP New Reno flow.



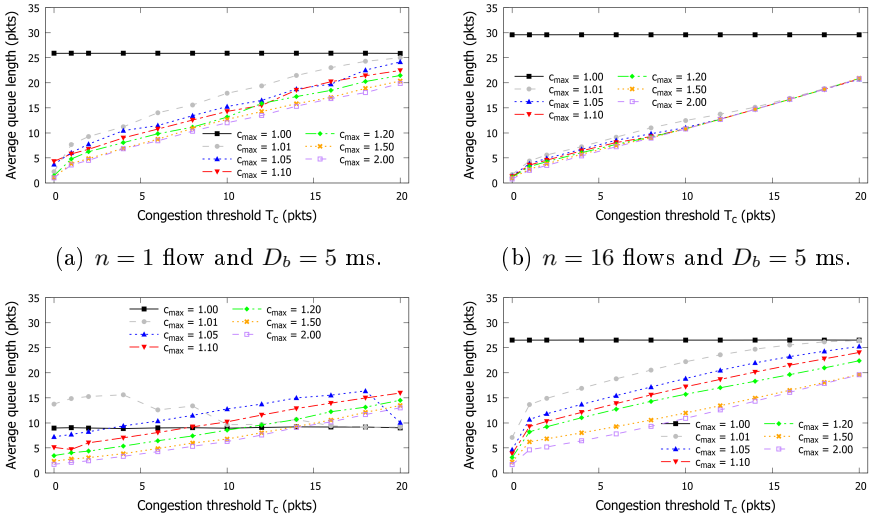
(b) $n = 16$ TCP New Reno flows.

Fig. 4. Link utilization on the bottleneck link for a one-way delay of $D_b = 50$ ms.

$n = 16$ TCP New Reno connections, a sufficiently large maximum congestion of $c_{max} \geq 1.05$ is needed in addition to keep the average queue length low.

The link utilization is exactly or close to 100% for almost all experiments with a $D_b = 5$ ms. This is different for $D_b = 50$ ms. Figures 4(a) and 4(b) provide the link utilization for $D_b = 50$ ms and $n \in \{1, 16\}$ TCP Reno flows. For $n = 1$ flow, the link utilization was only 92.5% with tail-drop ($c_{max} = 1$), but the values for CP-AQM fall below that level (80% – 90%), in particular for small congestion thresholds T_c and large maximum congestion c_{max}. Under these challenging conditions low average queue lengths come at the expense of reduced link utilization. For $n = 16$ flows, significantly reduced link utilization can be avoided by choosing the maximum congestion as $c_{max} \leq 1.05$.

Experiments with TCP Cubic Figures 5(a)–5(d) show that the more aggressive TCP Cubic variant leads to very similar results regarding average queue



(a) $n = 1$ flow and $D_b = 5$ ms.

(b) $n = 16$ flows and $D_b = 5$ ms.

(c) $n = 1$ flow and $D_b = 50$ ms.

(d) $n = 16$ flows and $D_b = 50$ ms.

Fig. 5. Average queue lengths on the bottleneck link for TCP Cubic.

length as TCP New Reno. For most configurations of T_c and c_{max}, the average queue length is slightly larger than for TCP New Reno. For $D_b = 50$ ms and $n = 1$ TCP Cubic flow, CP-AQM may even lead to longer average queue lengths for large T_c and small c_{max} than for tail-drop ($c_{max} = 1$). This looks counterintuitive at first sight, but may be due to implementation specifics of TCP Cubic and the fact that tail-drop regularly fills the entire queue before packet loss occurs.

We do not show figures for the link utilization with TCP Cubic and $D_b = 50$ ms, but report results. For $n = 1$ flow, the link utilization is about 98% like for tail-drop. For small congestion thresholds of $T_c \leq 5$ packets and large maximum congestions of $c_{max} \geq 1.2$, the utilization may range between 94% and 98%.

4.5 Recommendations for Configuration

We recommend to set the congestion allowance parameters R_{CA} and B_{CA} like proposed in Section 3.

Looking at all investigated networking scenarios, we propose to set $T_c = 5$ and $c_{max} = 1.2$ because that limits the average queue length to at most 12 packets. The price to pay is a reduced link utilization of 95% for $D_b = 50$ ms and $n = 16$ TCP New Reno connections and even lower for $n = 1$ TCP New Reno flow. When choosing a lower maximum congestion of $c_{max} = 1.05$, New Reno achieves almost full link utilization for $D_b = 50$ ms and $n = 16$ flows, but average queue lengths may reach 15 packets in that case. A larger maximum

congestion of $c_{max} = 1.5$ may limit the average queue length to 9 packets with some more decrease in utilization for $D_b = 50$ ms (82% for $n = 1$ flows TCP New Reno, 94% for $n = 16$ flows TCP New Reno, 96% for $n = 1$ flow and TCP Cubic, 99% for $n = 16$ flows TCP Cubic), which is moderate for TCP Cubic. Thus, there is a tradeoff between low average queue length and high utilization where a decision needs to be taken depending on preference.

5 Conclusion

We have presented CP-AQM as a new AQM mechanism based on the idea of congestion policing (CP). Two configuration parameters define congestion based on the state of the queue and two more configuration parameters define the behavior of the congestion policer which is based on a token bucket. We derived appropriate token bucket parameters and performed experiments to find suitable values for the remaining two parameters that define the congestion function.

To that end, we simulated average queue length and utilization in the presence of tail-drop and CP-AQM for various configurations, networking scenarios and transport protocols on a 10 Mb/s link. CP-AQM keeps the average queue length very short in case of persistent overload through non-responsive traffic. With reasonable configuration it achieves also short average queue lengths for TCP traffic (New Reno and Cubic) and 100% utilization if multiple flows are transmitted. The reduction is significant: CP-AQM ($T_c = 5$ packets) leads to an average queue length of only 7 packets for 16 TCP Cubic flows and a one-way delay of 5 ms instead of 30 packets for tail-drop. If only a single flow is transmitted, CP-AQM causes slightly decreased link utilization for large one-way delays in the range of 50 ms. The most intriguing feature of CP-AQM is that it keeps queue lengths very short for persistent non-responsive traffic while allowing mostly full utilization for TCP traffic. Moreover, our proposed configuration of CP-AQM assures that the entire buffer can be utilized by a single burst.

While this first study of CP-AQM is promising, it is unclear whether CP-AQM can be extended to cope with varying bandwidth to make it applicable also for wireless networks. CP-AQM should be compared to other AQM mechanisms such as RED [13], CoDel [28], PIE [30], and GSP [25]. A comparison is needed also for bandwidths other than 10 Mb/s, for more complex traffic patterns, and for other objectives like keeping the queue length very short even at the expense of significantly reduced utilization.

References

1. Adams, R.: Active Queue Management: A Survey. IEEE Communications Surveys & Tutorials 15(3), 1425–1476 (2013)
2. Allman, M.: Comments on Bufferbloat. ACM SIGCOMM Computer Communication Review 43(1), 30 – 37 (Jan 2013)
3. Baillargeon, S., Johansson, I.: ConEx Lite for Mobile Networks. In: Capacity Sharing Workshop (CSWS) (2014)
4. Baker, F., Fairhurst, G.: RFC7567: IETF Recommendations Regarding Active Queue Management (Jul 2015)

5. Baker, F., Pan, R.: RFC7806: On Queuing, Marking, and Dropping (Apr 2016)
6. Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., Zhang, L.: RFC2309: Recommendations on Queue Management and Congestion Avoidance in the Internet (Apr 1998)
7. Briscoe, B., Sridharan, M.: Network Performance Isolation in Data Centres using Congestion Policing. http://tools.ietf.org/html/draft-briscoe-conex-data-centre (Feb 2014)
8. Briscoe, B.: Re-feedback: Freedom with Accountability for Causing Congestion in a Connectionless Internetwork. PhD thesis, Department of Computer Science, University College London (2009)
9. Briscoe, B., Brunstrom, A., Petlund, A., Hayes, D., Ros, D., Tsang, I.J., Gjessing, S., Fairhurst, G., Griwodz, C., Welzl, M.: Reducing Internet Latency: A Survey of Techniques and their Merits. IEEE Communications Surveys & Tutorials 18(3), 2149 – 2196 (2016)
10. Briscoe, B., Jacquet, A., di Cairano-Gilfedder, C., Salvatori, A., Soppera, A., Koyabe, M.: Policing Congestion Response in an Internetwork using Re-feedback. In: ACM SIGCOMM. Portland, OR (Aug 2005)
11. Briscoe Ed., B., Woundy Ed., R., Cooper Ed., A.: RFC6789: Congestion Exposure (ConEx) Concepts and Use Cases (Dec 2012)
12. Cloonan, T., Allen, J., Cotter, T., Widrevitz, B., Howe, J.: Minimizing Bufferbloat and Optimizing Packet Stream Performance in DOCSIS 3.0 CMs and CMTSs. In: Cable-Tec EXPO 13. Atlanta, GA (Oct 2013)
13. Floyd, S., Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking 1(4), 397–413 (Aug 1993)
14. Gettys, J., Nichols, K.: Bufferbloat: Dark Buffers in the Internet. ACM Queue 9(11) (Nov 2011)
15. Gong, Y., Rossi, D., Testa, C., Valenti, S., Taht, M.D.: Fighting the Bufferbloat: on the Coexistence of AQM and Low Priority Congestion Control. In: IEEE INFOCOM Workshop on Traffic Measurement and Analysis (2013)
16. Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., Dumazet, E.: RFC8290: The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm. https://tools.ietf.org/html/rfc8290 (Jan 2018)
17. Hohlfeld, O., Pujol, E., Ciucu, F., Feldmann, A., Barford, P.: BufferBloat: How Relevant? A QoE Perspective on Buffer Sizing. Tech. Rep. 2012-11, TU Berlin, Faculty of Electrical Engineering and Computer Science (Nov 2012)
18. IETF Working Group on Active Queue Management and Packet Scheduling (AQM): Description of the Working Group. http://tools.ietf.org/wg/aqm/charters (2013)
19. IETF Working Group on Congestion Exposure (CONEX): Description of the Working Group. http://tools.ietf.org/wg/conex/charters (2010)
20. Jacquet, A., Briscoe, B., Moncaster, T.: Policing Freedom to Use the Internet Resource Pool. In: Re-Architecting the Internet (ReArch). Madrid, Spain (Dec 2008)
21. Jiang, H., Liu, Z., Wang, Y., Lee, K., Rhee, I.: Understanding Bufferbloat in Cellular Networks. In: Workshop on Cellular Networks: Operations, Challenges, and Future Design (CellNet) (Aug 2012)
22. Khademi, N., Ros, D., Welzl, M.: The New AQM Kids on the Block: An Experimental Evaluation of CoDel and PI. In: IEEE Global Internet Symposium (2014)
23. Kuhn, N., Lochin, E., Mehani, O.: Revisiting Old Friends: Is CoDel Really Achieving What RED Cannot? In: Capacity Sharing Workshop (CSWS) (2014)

24. Kutscher, D., Mir, F., Winter, R., Krishnan, S., Zhang, Y., Bernados, C.J.: Mobile Communication Congestion Exposure Scenario. http://tools.ietf.org/html/draft-ietf-conex-mobile (Oct 2015)
25. Lautenschlaeger, W.: Global Synchronization Protection for Packet Queues. http://tools.ietf.org/html/draft-lauten-aqm-gsp (May 2016)
26. Martin, J., Hong, G., Westall, J.: Managing Fairness and Application Performance with Active Queue Management in DOCSIS-based Cable. In: Capacity Sharing Workshop (CSWS) (2014)
27. Nichols, K., Jacobson, V., McGregor, Ed., A., Iyengar, Ed., J.: RFC8289: Controlled Delay Active Queue Management. https://tools.ietf.org/html/rfc8289 (Jan 2018)
28. Nichols, K., Jacobson, V.: Controlling Queue Delay. ACM Queue 10(5) (May 2012)
29. Pan, R., Natarajan, P., Baker, F., White, G.: RFC8033: Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem. https://tools.ietf.org/html/rfc8033 (Feb 2017)
30. Pan, R., Natarajan, P., Piglione, C., Prabhu, M.S., Subramanian, V., Baker, F., VerSteeg, B.: PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem. In: IEEE Workshop on High Performance Switching and Routing (HPSR) (2013)
31. Showail, A., Jamshaid, K., Shihada, B.: WQM: An Aggregation-Aware Queue Management Scheme for IEEE 802.11n Based Networks. In: Capacity Sharing Workshop (CSWS) (2014)
32. Sivaraman, A., Winstein, K., Subramanian, S., Balakrishnan, H.: No Silver Bullet: Extending SDN to the Data Plane. In: ACM Workshop on Hot Topics in Networks (HotNets) (Nov 2013)
33. Varga, A.: INET-2.2 released. http://inet.omnetpp.org/ (Aug 2013)
34. Varga, A.: OMNeT++ 4.3.1 released. http://www.omnetpp.org/ (Sep 2013)
35. Wand, S.: Network Simulation Cradle. http://research.wand.net.nz/software/nsc.php (2012)
36. White, G., Pan, R.: RFC8034: Active Queue Management (AQM) Based on Proportional Integral Controller Enhanced (PIE) for Data-Over-Cable Service Interface Specifications (DOCSIS) Cable Modems. https://tools.ietf.org/html/rfc8034 (Feb 2017)
37. White, G.: Active Queue Management in Docsis 3.X Cable Modems. Tech. rep., Cable Television Laboratories, Inc. (May 2014)