applied sciences

MDPI

Article

On Cross-Layer Interactions for Congestion Control in the Internet

Agnieszka Piotrowska

Department of Computer Networks and Systems, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland; agnieszka.piotrowska@polsl.pl

Abstract: Two key mechanisms of the Internet are congestion control in the Transmission Control Protocol (TCP) and Active Queue Management (AQM) in routers. The former divides the bandwidth between flows and prevents the Internet from congestion collapse. Simultaneously, the latter informs hosts of the forthcoming congestion by preventive dropping of packets in network nodes. Although these two key mechanisms may severely interact with each other, they are often being researched independently, in parallel. This has led to the development of a few new congestion controls and AQM algorithms known for excellent performance under the assumption that the counterpart remains unaltered. It is unclear, however, how these new solutions in both areas interact with each other. The purpose of this paper is to fill this gap. Namely, in an extensive set of simulations, the impact of interactions between the state-of-the-art congestion control and AQM algorithms on the TCP connection performance is studied. As a result, recommendations for using some particular TCP-AQM pairs, which are observed to perform especially well, are formulated.

Keywords: TCP; congestion control; Active Queue Management; New Reno; cubic; compound; CoDel; PIE; performance evaluation; cross-layer interactions

Citation: Piotrowska, A. On Cross-Layer Interactions for Congestion Control in the Internet. *Appl. Sci.* **2021**, *11*, 7808. https://doi.org/10.3390/app11177808

Academic Editor: Christos Bouras

Received: 21 July 2021
Accepted: 22 August 2021
Published: 25 August 2021

1. Introduction

Congestion is an inevitable phenomenon in many wired and wireless networks as network resources are limited in the available bandwidth. Congestion leads to dropping packets, unreliable transmission, high latency, and small goodput, which causes user frustration and the inability to use the Internet facilities effectively. On the other hand, sending data at a very low rate does not provide acceptable resource utilization and results in a poor user experience. Therefore, every user must send data at the "correct" rate to effectively use the available resources concerning other users. Altogether, this is the main role of congestion control implemented in TCP .

Given that it is impossible to know the capacity and availability of all links between every pair of sender and receiver, dealing with congestion is a critical mechanism for the proper functioning of the Internet. Congestion control is built on the received acknowledgments of the transmitted data. If packets are dropped and not acknowledged, it signals the sending rate is too high. If packets are properly delivered, the sending rate may be lower than the available bandwidth. Basing on this information, the sender can adjust the sending rate to maintain an accurate pace.

TCP remains the only transport protocol that controls the amount of sent data. Its main role is to control the transmission rate to maximize the utilization of the available bandwidth and respond to emerging congestion, provide reliable data delivery, and provide a fair share of link capacity to all competing flows. To maximize throughput, TCP increases the sending rate to the point it causes congestion and dropping of packets, leading to slowing down and recovery from loss. Therefore, designing a TCP behavior is always a problem of finding the optimal formula for a congestion window size that provides a favorable resource utilization yet does not overwhelm the network.

Since the first version of TCP in the 1970s [1], many TCP flavors have been introduced to adapt to the emerging problems, enhanced network capabilities, and varying patterns of user traffic characteristics. With the advancement in network technology, TCP was confronted with numerous problems such as under-utilization of bandwidth, unfairness, unnecessary retransmissions, and non-congestion losses.

TCP variants may be classified based on their behavior: loss-based (reactive), delay-based (proactive), and loss-based with bandwidth estimation or problem-orientated (congestion, reordering, wireless transmission, bandwidth utilization, and fairness issues). The detailed TCP survey with an analysis of strengths and weaknesses and classification of congestion control mechanisms is presented in [2]. Although several dozens of TCP versions exist, only a few are commercially implemented and used. This paper focuses on the TCP versions implemented in currently used operating systems and, according to [2], the most commonly used, i.e., TCP New Reno, TCP Compound, and TCP Cubic.

Most TCP variants lead to persistently full buffer in network nodes, which causes many undesired problems like buffer overflow and bufferbloat (see Section 4 for more details). The problem of buffer overflow is discussed since the early 1980s. In 1998, IETF indicated that the Congestion Avoidance mechanism of TCP, although powerful and essential, should be complemented with Active Queue Management (AQM) in network nodes [3]. The main role of AQM is to predict the incoming congestion and preventively drop some packets to reduce the transmission rate of TCP sources. The AQM should provide high link utilization, along with short and low varying queuing delays, which requires a low and stable queue in the first place.

Since 2015, AQM is recommended by IETF feature in all routers [3]. AQM, along with TCP/IP protocols, are fundamental to closed-loop congestion control and are the main elements of the network infrastructure. Although the problem of reducing and stabilizing end-to-end latency is known and has been discussed for many years, the widespread deployment of any AQM has not happened so far, mainly because of their sensitivity to varying network conditions. Some relatively new AQM schemes, i.e., PIE and CoDel, attempt to tackle the problem of tedious configuration and parametrization. Despite the considerable amount of research on TCP congestion control algorithms and AQM performance, the interdependence of those two mechanisms is yet to be efficiently studied and addressed. The unique characteristics of any congestion avoidance component require a comprehensive analysis under various network conditions.

In this paper, the interactions between the most commonly used TCP versions and state-of-the-art Active Queue Management algorithms are evaluated. We are trying to answer the following questions:

- To what extent an AQM influences the end-to-end congestion control mechanism?
- What are the effects of network parameters, such as RTT variation and congestion level, on the combined TCP and AQM performance?
- Can cross-layer interaction between TCP and AQM overcome the challenges of network heterogeneity and variability?
- Is there one universal TCP-AQM combination for all cases?

The rest of the paper is organized as follows. Section 2 reviews the related research. In Sections 3 and 4, the evaluated TCP and AQM approaches are described in detail. The performance evaluation of different AQM and TCP combinations in different scenarios is presented in Section 5. Finally, conclusions are provided in Section 6.

2. Related Work

The problem of congestion control is widely discussed in many research papers. The majority of them are focused on a specific key component of the Internet mechanisms evaluated with the assumption that other key components remain unchanged. Moreover, the presented analyses are usually oriented towards a specific purpose, e.g., the buffer size influence or impact of the network heterogeneity on different aspects of transmission performance.

Suffice it to mention several papers related to the evaluation of TCP versions assessed in this paper in conjunction with the Drop Tail (DT) mechanism. The authors of [4] explore the fairness of high-speed TCPs. The broader comparison of high-speed TCPs over the high-BDP network is drawn by Alrshah et al. [5]. In [6], the authors study the interactions between multiple versions of TCP. Another comparison of multiple TCP versions (Cubic and New Reno) in various network scenarios (wired/wireless) is presented in [7]. The authors of [8] perform a similar evaluation concerning Linux versions of TCPs. The analytical tool to compare the TCP versions evaluated in this paper is submitted in [9]; the simulation results and performance analysis are reported in [10].

In this paper, we focus on two AQMs: PIE and CoDel. The comparison of those state-of-the-art AQMs' limits and hints for tuning and improvement are presented in [11]. The authors also provide a discussion concerning the operational range of both algorithms. The assessment of the introduced delay and bandwidth utilization provided by three different AQMs, namely, ARED, PIE, and CoDel for the sender using TCP Cubic, in the presence of CBR-traffic, is studied in [12]. Different queuing mechanisms, in the presence of TCP Reno, are also evaluated in [13].

A few papers evaluate the cross-layer interaction between end-to-end TCP congestion control and buffer management strategies provided by AQM. In [14], the performance of AQMs in the presence of new versions of the TCP congestion control mechanism is evaluated. The paper focuses on TCP SACK, New Reno, FACK, and Cubic versions and compares RED, AVQ, PI, REM, and AN-AQM along with DT approach. All the evaluated AQM algorithms, although thoroughly discussed and analyzed in many research papers, are not commonly used in commercial buffers due to configuration issues; they do not fall into the category of recommended by RFC 7567 [3] AQMs algorithms. In this paper, the recommended CoDel and PIE are verified against the most widespread TCP versions.

The authors of [15] evaluate RED, PIE, and CoDel, however in combination with TCP Reno, BIC, and Vegas, which are rather less popular TCP flavors. Moreover, the authors use a very simple, real-time testbed consisting of a client, router, and server. The client sends data to the server via the router, and the receiver sends back ACKs via the same router. The results are obtained based on one generated flow. The outcomes presented in this paper are based on a much more diversified and extensive set of simulation scenarios and more common TCP versions.

In the assumptions, those nearest to this paper's work are presented in [16]. The authors perform a cross-comparison of eight TCP flavors and five AQM variants in several network scenarios. The most significant differences concern network configuration, i.e., link capacity, MTU, as well as the selection of RTT distribution. The authors restrain the bottleneck link to 10 Mbps and MTU to 500 B. Assuming TCP Cubic and Compound were designed for more demanding environments (high speed and high latency) in 10 Mbps simulation, their expected superiority may not be visible. Moreover, the smaller MTU shortens the experienced delay in low-speed networks, however, leads to network under-utilization, especially in high-speed environments. In this paper, the TCP-AQM interactions are evaluated in the Data Center scenario with 1 Gbps links and the Access Link scenario with bandwidth set to 100 Mbps. In both scenarios, a 1500 B MTU is used.

RTT proposed in this paper is based on the recommendations for Data Center and Access Link scenarios provided in [17]. The selected values are supposed to provide a "realistic distribution of round-trip times for traffic on the central link". In the aforementioned paper, the authors use their own uniformly distributed proposition of RTT settings.

Summarizing, the novelty of this paper lies in the following: (1) comparison of TCP-AQM pairs in classic network scenarios, (2) evaluation of the most common combinations of TCPs and modern AQMs, and (3) verification of how the most widespread TCP versions interact with each other in combination with AQMs in representative network scenarios.

3. TCP Variants

Not all proposed in the literature TCP versions are available in common operating systems. Only a few of them gained popularity and are provided for everyday usage. In this section, the most popular and widespread TCP versions are recalled.

3.1. TCP New Reno with SACK

TCP New Reno is implemented in most operating systems, along with SACK (Selective Acknowledgment) support. It is the default version used by Windows 10 and Windows Server 2016 operating systems and FreeBSD.

TCP New Reno is an improved version of TCP Reno [18]. It modifies only the Fast Recovery phase, while other phases are identical to TCP Reno, which is based on one of the earliest versions of TCP, i.e., TCP Tahoe proposed by Jacobson [19]. The congestion window of TCP New Reno increases exponentially in the Slow Start phase and linearly in the Congestion Avoidance phase. When duplicated ACKs are received, the congestion is detected, the congestion window is reduced by half, and the Fast Recovery phase is started.

TCP New Reno overcomes the problem of multiple losses, which causes reducing the congestion window multiple times. TCP New Reno remains in the Fast Recovery phase until all outstanding segments, at the time it enters the Fast Recovery, are acknowledged. During this phase, it may retransmit multiple loss segments before returning to the Congestion Avoidance phase.

Without SACK, New Reno requires one RTT to detect each packet loss. SACK allows acknowledging the received packets selectively; therefore, it improves the recovery from multiple losses. Each ACK has a block that describes which segments have already been received and do not require retransmission.

The main flaw of TCP New Reno is its poor bandwidth utilization in high-speed environments with moderate or high RTTs, e.g., in a network having 10 Gbps capacity, 100 ms RTT, and a maximum packet size of 1500 bytes, it would take almost two hours of error-free transmission to get to a theoretical upper bound of transfer rate [20].

3.2. TCP Cubic

TCP Cubic [21] has been the default congestion control algorithm in Linux, Android, and iOS since 2018, and it is also the default version in Windows Server 2019.

The main design objective of TCP Cubic is to provide effective bandwidth utilization in high-speed scenarios and provide RTT and inter-TCP fairness to other competing flows. The main idea is to slow down the increase of the congestion window ($cwnd$) when the previous network maximum is achieved (concave increase) and increase it rapidly if the threshold is surpassed without a loss (convex increase). This is achieved using a cubic function for the congestion window size (instead of linear):

$$cwnd = C * (\delta - \sqrt{\beta * cwnd_{max}/C})^3 + cwnd_{max}, \quad (1)$$

where C is a predefined constant, δ is the time elapsed since the last congestion event, β is a coefficient of the multiplicative decrease in the Fast Recovery phase, and $cwnd_{max}$ is the observed congestion window size just before the last registered loss. The function of the congestion window growth results in a very fast rise in the transmission rate after loss detection and entering the Fast Recovery phase. The closer to the target value of the congestion window $cwnd_{max}$, the slower increase in the growth rate. TCP Cubic improves scalability and stability under fast and long-distance networks. It ensures that the achieved throughput is not lower than the throughput achieved by TCP New Reno by providing that $cwnd_{max}$ is set to $cwnd_{reno}$ in the case it would be lower when calculated using Equation (1).

TCP Cubic is based on BIC TCP (Binary Increase Congestion control) [22], which extends TCP New Reno with an additional phase meant to rapidly discover, in a binary search manner, the optimal congestion window value. The main advantage of Cubic is its RTT-independent congestion window growth function, which improves the RTT fairness.

3.3. TCP Compound

TCP Compound (C-TCP) [23] is a Microsoft algorithm. It was implemented in Windows XP (as a patch), Vista, and Windows 7. It is also available in Windows 10 and Windows Server 2016, as well as in the Linux kernel. According to the work in [2], it was the most deployed version of TCP in 2010, owing to the fact that it has replaced the conventional congestion control in Windows operating systems since Windows Vista and Windows 2008 Server. However, in Windows 10, the default congestion control was set to New Reno and Windows Server 2019 uses Cubic by default, therefore its commonness dropped.

The first objective of TCP Compound is to fully utilize the available bandwidth in high-speed and long-distance networks (high BDP) and provide RTT- and inter-TCP fairness at the same time. The algorithm uses a delay-based estimator of the network state (based on TCP Vegas [24]) to combine the Reno-type congestion control with a scalable component of congestion control derived from High-Speed TCP (HS-TCP) [20].

The congestion window increases exponentially in the Slow Start phase, identical to TCP Reno Slow Start. In the Congestion Avoidance phase, the window size (win) is the sum of the standard congestion window ($cwnd$), calculated as in Reno and the additional parameter ($dwnd$), which is calculated as follows:

$$dwnd(t+1) = \begin{cases} dwnd(t) + (\alpha * win(t)^k - 1), & \text{if } \Delta < \gamma, \\ dwnd(t) - \eta * \Delta, & \text{if } \Delta \geq \gamma, \\ win(t) * (1 - \beta) - cwnd/2, & \text{if loss is detected,} \end{cases} \quad (2)$$

where η and γ are predefined constants, α is an increase coefficient in the Congestion Avoidance phase, and β is a decrease factor after a loss detection. Δ is the Vegas estimator of network buffering. If it is below γ, the congestion window increases according to the HS-TCP rule. If it exceeds the threshold, the additional component is gently reduced to provide a smooth transition between the aggressive behavior of HS-TCP and the much slower Reno mode.

The main flaw of C-TCP is inherited from TCP Vegas and affects the accuracy of the estimator. If a competing flow observes the minimal RTT value that already consists of a buffering delay caused by an already existing flow, it behaves much more aggressively and unfairly to other flows.

4. AQM Algorithms

Packet buffers are essential elements of a network router. They compensate for the traffic bursts, which are a natural consequence of the best-effort nature of the network. However, the unwelcome consequence of buffering is an additional packet delay, which affects the performance of interactive and transaction-based applications. High and variable latency may also impair the congestion control mechanism of TCP. The main problems with buffers are buffer overflows and so-called bufferbloat, i.e., "persistently full buffer" [25]. Buffer overflows are caused by bottlenecks that result from a decrease in the transmission rate or at the convergence point of many links.

Throughout the last 20 years, despite the awareness and dramatic increase in the buffer memory size and memory access speed up, the problem still affects network performance. Actually, the increase of memory size and availability is the main cause of bufferbloat, and the solution for this is AQM. With the increased usage of the Internet as well as the change in traffic characteristics due to the growth of video streaming and time-critical traffic (video conferences, VoIP, and gaming), the urge for proper buffer management is even more critical.

The main limitation of most AQM algorithms is that they are vulnerable to parameters' tuning and require adapting of parameters when the network condition changes. This is probably the most important reason why AQMs (as RED, for example), although thoroughly researched, are not commonly deployed. Drop Tail is still the most popular

buffer management strategy applied at routers, resulting in packet drops only when the buffer is full.

The purpose of any Active Queue Management algorithm is to control the queue size to reduce its variability and queuing delay and preserve the possibility of absorbing packet bursts at the same time. An extensive survey of Active Queue Management schemes, developed and evaluated in the last decades, has been published by Adams [26]. Two state-of-the-art AQM algorithms that address the problem of bufferbloat along and provide a parameterless configuration are presented below.

4.1. PIE

The main purpose of the Proportional Integral Enhanced (PIE) controller [27] is to deal with a bufferbloat problem by effectively controlling the average queuing delay to a target value and providing high bandwidth utilization under varying conditions at the same time. PIE is an enhancement to the Proportional Integral (PI) controller. Assumptions that stand behind PIE design are straightforward implementation and scalability in high-speed networks. In contrast to PI, PIE parameters are auto-tuned.

On a reception, PIE drops a packet randomly with a probability p before enqueuing. When the packet is dequeued, PIE calculates the average dequeue rate avg_drate. This value is used to estimate the available bandwidth and to track its variation. The calculations are performed in measurement cycles. If the current queue length exceeds a defined threshold (set to 16 kB by default), a new measurement cycle is started, during which PIE keeps track of the summed size of dequeued packets. The threshold should mean that the measurements are carried out when the queue backlog is large enough; such a precaution provides a meaningful estimate of the dequeue rate. The measurement cycle ends when the number of dequeued bytes exceeds the defined threshold. Then, PIE computes the dequeue rate and derives the average dequeue rate as a weighted average of the previous average dequeue rate and the last dequeue rate sample.

For every T_{update} interval rate (15 ms as recommended by [27]), the current queuing delay is updated—from the Little's Law—as a ratio of the current queue length $qlen$ and the estimated average dequeue rate avg_drate. The dropping probability p is also updated as a linear combination of the previous value, the difference between the current queue delay cur_{del} and the reference value ref_{del} (weighted by a parameter α), and the difference between the current queue delay cur_{del} and the previous queue delay old_{del} (weighted by a parameter β). PIE steady state is achieved when the current queue delay equals the reference queue delay. PIE scales parameters α and β to adapt the dropping probability to the current network conditions. Summarizing, in every rate interval, the following calculations occur:

$$\begin{aligned} cur_{del} &= \frac{qlen}{avg\_drate}, \\ p &= p + \alpha * (cur_{del} - ref_{del}) + \beta * (cur_{del} - old_{del}), \\ old_{del} &= cur_{del}, \end{aligned} \tag{3}$$

Basing on the current value of dropping probability p, the scale parameters α and β are determined as follows:

$$\begin{aligned} \text{if } p < 0.01, \alpha &= \overline{\alpha}/8, & \beta &= \overline{\beta}/8, \\ \text{else if } p < 0.1, \alpha &= \overline{\alpha}/2, & \beta &= \overline{\beta}/2, \\ \text{else, } \alpha &= \overline{\alpha}, & \beta &= \overline{\beta}, \end{aligned} \tag{4}$$

where $\overline{\alpha}$ and $\overline{\beta}$ are the initial values of both parameters.

The main problem of PIE in the real world comes from the implementation. Linux traffic control module dequeues several packets simultaneously, which influences the average dequeue rate, which is closer to the estimated rate at which packets are stored in

the device driver's transmission ring. To overcome this issue, the authors of [28] proposed using packet timestamps for dequeue rate calculations. Moreover, in [29], the authors claim that in 100 Mbps and faster networks, adapting the ref_{del} results in shorter standing queues.

4.2. CoDel

The CoDel (Controlled Delay) algorithm was first presented in [25] and was published as RFC 8289 document [30]. Its main purpose is to control and provide low delays while permitting bursts of traffic. The superiority of CoDel over other AQMs is also the lack of configuration. It is parameterless and insensitive to varied round-trip delays, link rates, and traffic loads.

Implementation of CoDel is based on the idea of distinguishing a good queue that allows maintaining the 100% link utilization from a bad queue, resulting from setting a congestion window over the pipe size, which persists in a queue for several RTTs. As long as the good queue is acceptable and beneficial, the bad queue should be avoided.

Many AQMs use the average queue length as an indicator of congestion. However, the metric is sensitive to the process of estimation and measurement. CoDel consists of three main components: an estimator, a set point, and a control loop.

CoDel tries to distinguish the bad queue from the good queue. The good queue may be ignored as it will disappear in the RTT whatsoever. The bad queue should be handled. The distinction is made based on the delay. CoDel uses sojourn time to estimate the congestion. If there is at least one packet with zero sojourn time, the queue is not persistent (it is a good queue). The packet sojourn time is measured when the packet is dequeued for transmission.

CoDel switches between a dropping state (bad queue detected) and a non-dropping state. It starts in the non-dropping state. The persistent queue is detected by tracking the minimum experienced packet delay in intervals. The sojourn time is compared to the target value (5 ms by default), and if it is above, a timer is started. As long as the noticed sojourn times remain above the target value, the timer is running. If it reaches the value of the interval ($\lambda = 100$ ms by default), CoDel enters the dropping state.

During the dropping state, CoDel drops a packet and resets the interval timer; the number of dropped packets is tracked using the parameter N_{drop}. The dropping and resetting of the timer are repeated until the sojourn time gets below the target value. When a single packet is dropped, the interval value is reduced by the square root of the number of drops, as follows:

$$\mu = \frac{\lambda}{\sqrt{N_{drop}}}. \tag{5}$$

Thus, CoDel drops a packet only every μ to give senders time to reduce transmission rate.

When CoDel returns to the non-dropping state, it restores the default values for the target delay and the interval, while the number of dropped packets is set to zero. The set point target of 5% of nominal RTT (5 ms for 100 ms RTT) yields substantial utilization improvement for the small added delay. The predefined values of the target and the interval length were determined via experiments. In [25], the authors provide discussion and justification for their selection.

The main advantage of CoDel is its lack of configuration and easy implementation, as the only addition to packet arrival is the timestamp registered for each packet. However, the authors of [31] noticed that the default target value should be tuned "to be at least the transmission time of a single MTU-size packet at the prevalent egress link speed". In wireless low-speed access networks, it may be at least 15 ms.

5. Performance Evaluation

The purpose of the simulations was to verify how the most commonly used TCP congestion control algorithms interact with the state-of-the-art Active Queue Management approaches. Simulations were carried out using ns-3, release 3.30 with added TCP Cubic [32], and TCP Compound [33] implementations. The simulation network was configured based on the topology and scenarios presented in [17] and described in detail below.

The standard dumbbell topology with two routers and six nodes (N1–N6) was used (see Figure 1). The simulations were carried out using two basic scenarios presented in [17], i.e., Data Center and Access Link.

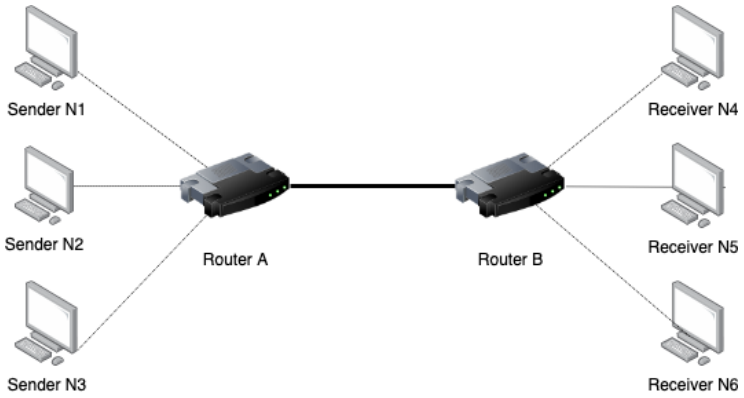


Figure 1. The dumbbell topology.

The Data Center scenario modeled a high-speed network with low delays. All links had a capacity of 1 Gbps. Links between nodes N1, N2, N4, and the router had 10 µs delay, links between nodes N3, N5, N6, and the router had 100 µs delay; the central link introduced no delay (0 ms). The average RTT of all nine connections was around 200 µs. The recommended buffer size should absorb 10 µs of traffic which, assuming 1500 B sized packets, was 900 packets.

In the Access Link scenario, every link rate was set to 100 Mbps, the propagation delay between routers A-B was set to 2 ms. The one-way propagation delays were distributed from 0 ms to 75 ms. The RTT values for all connections are gathered in Table 1. According to the work in [17], this is a representative distribution of RTTs on the central link. The buffer size was set according to the BDP rule (using the mean RTT), which was 900 packets.

Table 1. Values of RTT [ms] between all node pairs.

N1–N4	8	N2–N4	32	N3–N4	58
N1–N5	78	N2–N5	102	N3–N5	128
N1–N6	154	N2–N6	178	N3–N6	204

TCP sources were located in nodes N1–N3, nodes N4–N6 acted as data recipients. All TCP flows used 1500-byte packets. The total number of TCP flows was set according to the assumed congestion scenario. Three congestion scenarios were considered: the uncongested network (9 flows), mild congestion (90 flows), and heavy congestion (900 flows). Flows were started randomly during the first three seconds. Simulations were carried out for 100 s.

For every scenario, the simulations were performed for a single TCP version (common TCP) or all TCP versions simultaneously (mixed TCPs). In the common TCP scenario sets, the same TCP congestion control mechanism for all nodes was used throughout the simulation. Simulations were carried out for each TCP algorithm in combination with every buffer management strategy and every congestion state. In the mixed TCPs scenario

set, node N1 used TCP New Reno, node N2 used TCP Compound, and node N3 used TCP Cubic. Simulations were executed for each AQM and DT pair under varying congestion state.

Although PIE and CoDel should not require any configuration, according to the work in [29], even in 100 Mbps, the default reference value of delay ref_{del} set to 20 ms results in PIE acting as a simple DT, which was verified through the simulation. Therefore, in 1 Gbps and 100 Mbps networks, the target delay variable of PIE was set to 2 ms and 5 ms, respectively. Moreover, to provide comparable results for CoDel in the Data Center scenario, the sojourn time was also set to 2 ms.

To compare the obtained results, the overall throughput of the bottleneck link was monitored, as well as the mean queue length, the standard deviation of queue length, and the drop ratio at router A. The simulations lasted for 100 s, but the first 30 s were used for stabilization. Therefore, the queue measurements were taken from the 30th second of the simulation. The mean throughput for each flow was also calculated. Jain's index was computed to evaluate the fairness of link sharing between competing flows. The Jain's index calculated for different flow combinations is interpreted as inter-, intra-, or RTT-fairness. The inter-fairness indicated a fair share among flows using the same TCP congestion control mechanism. The intra-fairness concerned flows that use different TCP versions, and RTT-fairness applied to flows transmitted through links with different RTT values.

5.1. Data Center Scenario, Common TCP

The first simulation set is supposed to verify the overall performance and the cross-layer interactions between AQMs and TCP congestion control mechanisms in the Data Center scenario, i.e., with high-speed links and very short propagation delays, under the varying traffic load. The simulation results are gathered in the following tables. Table 2 presents the quartiles of single flows' throughput. Table 3 gives the Jain's index (inter-fairness) among all flows. Table 4 compares the average queue size with standard deviation and finally, Table 5 presents the packet drop ratio at the bottleneck link.

The overall throughput is very similar for all TCPs and AQMs interactions; it varies from 0.92 to 0.93 Gbps in the uncongested network to 0.95–0.96 Gbps during heavy congestion. Neither the AQM algorithm nor the TCP version influences the bandwidth utilization. It also implies that the average throughput of every TCP connection is similar and equal to 103, 10.5, and 1.05 Mbps in uncongested, mildly, and heavily congested networks, respectively. However, when comparing the median values as well as the first and the third quartile of the flow's throughput (see Table 2), the differences are significant.

First of all, regardless of the traffic load, the worst results, in terms of fairness, are observed for TCP Cubic and DT combinations. The median throughput is far below the average and, the flows' throughput distribution does not appear uniform. More precisely, during heavy congestion, DT with all TCPs is not able to provide acceptable fairness. It is not surprising, given the fact that only around 10 to 15% of all streams are able to transmit data packets effectively. Most connections are not even established due to continuous losses.

Replacing DT with PIE or CoDel notably improves the fairness and similarity of the mean throughput registered by every flow (Table 2). It is also visible when comparing the Jain's index (Table 3); in the majority of all interactions with PIE or CoDel, it is equal or almost equal to 1.00. However, it is worth mentioning that in the uncongested network, C-TCP with DT provides better results in terms of fairness and throughput variations than in combination with any AQM. Nonetheless, the higher the network load, the poorer inter-fairness yielded by C-TCP and DT.

Table 2. Quartiles of flows' throughput [Mbps] in the Data center scenario with common TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

		DT	PIE	CoDel
UC	NReno	56.72/83.54/145.79	100.83/102.17/105.78	98.32/99.55/108.67
	Cubic	49.91/84.06/138.36	96.60/100.70/109.61	98.67/99.84/108.67
	C-TCP	85.47/107.47/118.63	86.29/89.57/130.91	81.03/90.47/136.61
MC	NReno	2.40/5.65/14.15	9.98/10.45/10.85	10.02/10.44/10.82
	Cubic	0.00/0.04/23.41	9.93/10.26/10.89	10.07/10.46/10.84
	C-TCP	7.52/10.09/11.14	10.05/10.41/10.91	10.00/10.43/10.79
HC	NReno	0.00/0.00/0.00	0.95/1.05/1.14	0.98/1.06/1.14
	Cubic	0.00/0.00/0.00	0.94/1.04/1.16	0.94/1.05/1.17
	C-TCP	0.00/0.00/0.02	0.96/1.05/1.14	0.98/1.06/1.14

Table 3. Jain's index (fairness) among long-lived flows in the Data center scenario with common TCPs, UC—uncongested, MC—mildly congested, HC—-heavily congested network.

		DT	PIE	CoDel
UC	NReno	0.75	1.00	0.99
	Cubic	0.70	1.00	1.00
	C-TCP	0.97	0.96	0.94
MC	NReno	0.48	1.00	1.00
	Cubic	0.25	0.99	1.00
	C-TCP	0.88	1.00	1.00
HC	NReno	0.16	0.98	0.99
	Cubic	0.12	0.97	0.98
	C-TCP	0.19	0.98	0.99

In Table 4, the average queue size and standard deviation of the queue size are listed. The queue length variation over simulation time is depicted in Figure 2 (uncongested network), Figure 3 (during mild congestion), and Figure 4 (during heavy congestion).

The highest values of the average queue size are noticed for all DT cases since DT fills the entire buffer before dropping packets. Moreover, in the uncongested network, DT causes major queue size variations, which implies the high variability of experienced delay. PIE and CoDel significantly reduce the average queue size and its variability. In the uncongested network, the best results in terms of queue length and stability are observed for CoDel and C-TCP.

In the mildly congested network, both AQMs—PIE and CoDel—with any TCP combination, provide rather stable and much shorter than DT queues. There are no significant differences for any TCP-AQM pair. PIE in all TCP interactions yields shorter queues; actually, the average queue size of PIE is similar in all load scenarios. The average queue size provided by CoDel heavily depends on the congestion. Once again, in this load scenario, the most stable queue provides C-TCP and CoDel combination.

Table 4. Mean queue length and standard deviation of queue length [pkts] in the Data center scenario with common TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

		DT	PIE	CoDel
UC	NReno	814 (69)	169 (41)	173 (10)
	Cubic	859 (47)	168 (26)	172 (4)
	C-TCP	845 (39)	167 (23)	162 (14)
MC	NReno	873 (33)	166 (30)	222 (28)
	Cubic	900 (1)	167 (22)	216 (28)
	C-TCP	877 (17)	167 (24)	204 (18)
HC	NReno	899 (1)	168 (59)	724 (207)
	Cubic	899 (1)	168 (47)	716 (224)
	C-TCP	899 (8)	169 (52)	732 (214)

During heavy congestion, PIE results in notably smaller average queue size and much lower queue size variations than any other buffer management strategy. In Figure 4, it is visible that PIE allows filling the buffer at first but eventually stabilizes the queue around the target value. CoDEL with any TCP introduces enormous variability. The queue oscillates between the maximum and the target value, which results in high latency variations.

In Table 5, the summarized drop ratio at the bottleneck bandwidth is listed. In the uncongested network, the drop ratio is negligible. When the traffic load increases, the number of losses intensifies. It may appear that the best results in terms of packet loss ratio may be achieved when DT is used, but we have to keep in mind that in the mildly or heavily congested networks, only some part of all flows can perform an effective transmission. In all cases with any AQM, the drop ratio is comparable. Note that PIE with all TCPs causes a slightly higher drop ratio than CoDel and TCP Cubic with any AQM leads to more losses in comparison to New Reno or C-TCP.

Table 5. Packet drop ratio [%] in the Data center scenario with common TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

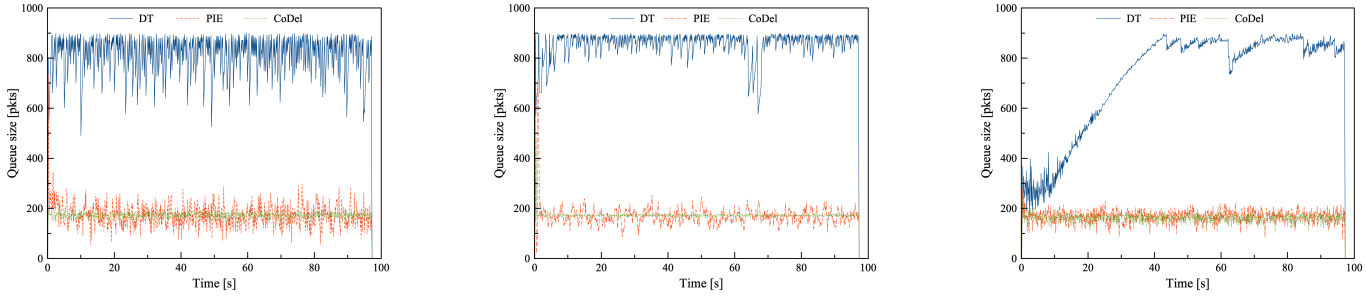
		DT	PIE	CoDel
UC	New Reno	0.02	0.08	0.06
	Cubic	0.03	0.06	0.06
	C-TCP	0.00	0.02	0.02
MC	New Reno	0.40	1.64	1.45
	Cubic	0.13	1.76	1.54
	C-TCP	0.34	1.56	1.43
HC	New Reno	0.67	6.53	5.78
	Cubic	0.48	6.98	6.46
	C-TCP	0.47	6.86	6.04

The presented results reveal that regardless of the applied TCP, there are problems that can be addressed by improving the buffer management strategy. As the congestion grows, CoDel has difficulty in maintaining a short and stable queue. The CoDel's sensitivity to network load was suggested in [11], and the above results confirm this dependence in a high-speed network scenario. CoDel allows filling the whole buffer because, during the dropping state, it drops a packet every μ, which is updated as per Equation (5) and under heavy congestion, it is not enough. To prevent this behavior, the update parameter should be tuned to provide faster response time in high-speed networks. The default value of λ is set to the value of nominal RTT, and in this scenario the average RTT is 500 times

smaller. However, it should be verified via simulation if tuning only the interval parameter is enough.

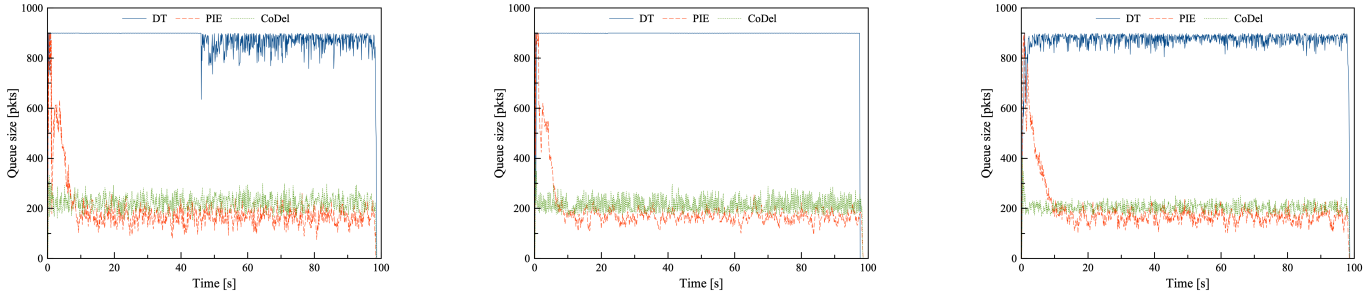
Summary:

- Replacing DT with PIE or CoDel considerably improves the achievable fairness, the experienced delay, as well as delay variations for all evaluated TCP congestion control mechanisms.
- PIE provides satisfactory and comparable results regarding fairness, drop ratio, and queue stability in all network load scenarios.
- TCP Compound provides satisfactory fairness and transmission efficiency even with DT in the uncongested and mildly congested network.
- **The following combinations provide the best results in terms of satisfactory fairness, delay, and drop ratio at the same time: CoDel with C-TCP (for uncongested and mildly congested network) and PIE with any TCP (during heavy congestion).**



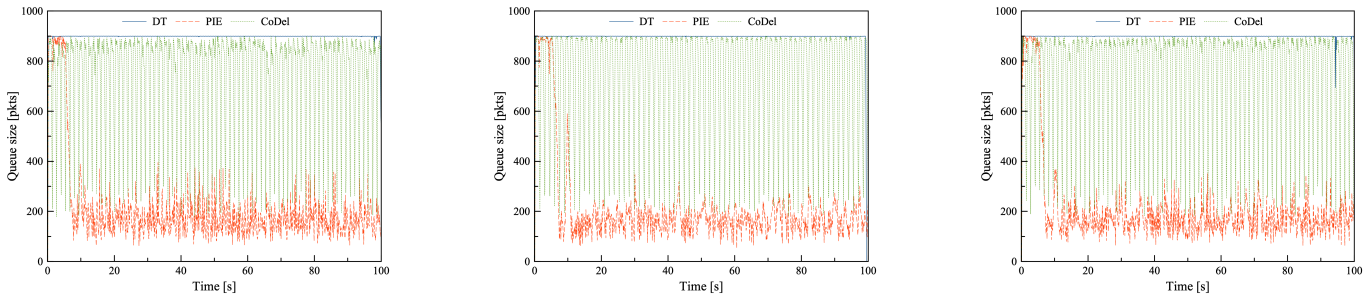
(**a**) New Reno (**b**) Cubic (**c**) C-TCP

Figure 2. The queue length of DT, PIE and CoDel in combination with (**a**) New Reno, (**b**) Cubic, and (**c**) C-TCP in the Data center scenario with common TCPs in uncongested network.



(**a**) New Reno (**b**) Cubic (**c**) C-TCP

Figure 3. The queue length of DT, PIE and CoDel in combination with (**a**) New Reno, (**b**) Cubic, and (**c**) C-TCP in the Data center scenario with common TCPs during mild congestion.



(**a**) New Reno (**b**) Cubic (**c**) C-TCP

Figure 4. The queue length of DT, PIE and CoDel in combination with (**a**) New Reno, (**b**) Cubic, and (**c**) C-TCP in the Data center scenario with common TCPs during heavy congestion.

5.2. Access Link, Common TCP

The second simulation group was supposed to verify the performance and interactions in a more common scenario with a different delay set to each link, representing the access link connecting an institution to an Internet Service Provider (ISP) [17]. The results are gathered in the following tables. Table 6 presents the quartiles of single flows' throughput. Table 7 compares Jain's index (RTT-fairness) for all flows. Table 8 gives the average queue size with its standard deviation, and Table 9 shows the packet drop ratio.

Introducing the varied RTT did not change the overall throughput. This is because neither the AQM algorithm nor the TCP version influences the bandwidth utilization, which is approximately 93–95%. It also implies that the average throughput of TCP connections is comparable in all cases. However, the throughputs of particular flows are significantly different. It is visible when we compare the quartiles of single flows' throughput (Table 6) and Jain's index (Table 7).

Table 6. Quartiles of flows' throughput [Mbps] in the Access Link scenario with common TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

		DT	PIE	CoDel
UC	New Reno	2.74/5.50/14.26	5.41/6.51/12.83	4.79/6.80/13.98
	Cubic	4.78/9.25/16.48	7.34/8.62/11.68	6.85/9.63/12.70
	C-TCP	8.28/10.63/11.64	5.67/10.93/14.88	6.31/11.57/13.42
MC	New Reno	0.53/0.78/1.06	0.48/0.70/1.20	0.52/0.74/1.17
	Cubic	0.00/0.01/1.93	0.68/0.82/1.14	0.72/0.90/1.16
	C-TCP	0.56/0.96/1.32	0.47/0.67/1.19	0.51/0.71/1.15
HC	New Reno	0.05/0.07/0.12	0.08/0.10/0.13	0.08/0.10/0.12
	Cubic	0.05/0.09/0.13	0.08/0.09/0.13	0.08/0.10/0.12
	C-TCP	0.06/0.08/0.12	0.08/0.10/0.13	0.08/0.10/0.12

Table 7. Jain's index (fairness) among long-lived flows in the Access Link scenario with common TCPs, UC–uncongested, MC—mildly congested, HC—heavily congested network.

		DT	PIE	CoDel
UC	New Reno	0.41	0.59	0.61
	Cubic	0.69	0.86	0.84
	C-TCP	0.94	0.82	0.89
MC	New Reno	0.54	0.61	0.65
	Cubic	0.24	0.76	0.80
	C-TCP	0.78	0.59	0.62
HC	New Reno	0.60	0.87	0.90
	Cubic	0.65	0.87	0.91
	C-TCP	0.68	0.89	0.91

Varied RTT highly influences how the bandwidth is shared among all long-lived flows. In the analogous DC scenario, all evaluated TCPs combined with both AQMs, provide very high fairness (Jain's index around 1); in this scenario, especially in uncongested and mildly congested networks, achieving even close results proved unattainable.

In the uncongested and mildly congested network, the worst results in terms of equal bandwidth sharing are observed for two interactions, i.e., New Reno and Cubic with DT; the median throughput is far below the average, the Jain's index is also low. Moreover, New Reno, even with PIE or CoDel, does not provide satisfactory results in terms of RTT-fairness. Similarly, as in the DC scenario with common TCPs (see Section 5.1), the C-TCP and DT combination yields the highest RTT-fairness. Note that Cubic, especially in the mildly congested network, benefits the most from replacing DT with PIE or CoDel. TCP Cubic

preserves higher fairness than the other two protocols because the congestion window growth function is independent of RTT.

In a heavily congested network, the fairness provided by any TCP combined with PIE or CoDel is significantly better than in combination with DT. Nevertheless, the values of Jain's index are noticeably worse than in the Data Center scenario, which implies that the varied RTT hinders providing a fair share of the bottleneck bandwidth.

Table 8. Mean queue length and standard deviation of queue length [pkts] in the Access Link scenario with common TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

		DT	PIE	CoDel
UC	New Reno	727 (142)	34 (27)	29 (23)
	Cubic	772 (85)	48 (26)	50 (18)
	C-TCP	167 (126)	23 (20)	25 (20)
MC	New Reno	832 (50)	54 (26)	94 (36)
	Cubic	899 (1)	52 (22)	91 (30)
	C-TCP	830 (46)	42 (17)	68 (18)
HC	New Reno	891 (10)	44 (21)	588 (272)
	Cubic	894 (7)	43 (20)	656 (280)
	C-TCP	893 (8)	45 (21)	592 (281)

The queue size and stability dependence on the TCP version and buffer management strategy are presented in Table 8. In the DC scenario, we could observe that the average queue size and its stability depend on the level of congestion and the buffer management strategy applied at the router. In this scenario, it is apparent that the TCP version also influences the queue behavior. In most cases, TCP Cubic leads to slightly higher average queue size in comparison to New Reno and C-TCP.

In the uncongested network, Drop Tail strategy with New Reno and Cubic leads to considerable average queue size and queue size variations, which implies the high variability of experienced delay. Surprisingly C-TCP, even with DT, is able to maintain a relatively short average queue size, however with high fluctuations. C-TCP with PIE or CoDel provides the best results in terms of the average queue length. All TCP versions benefit from interacting with PIE or CoDel in terms of the average queue size and the standard deviation of the queue size; all TCP-AQM pairs provide a rather low and stable queue (see Figure 5).

During mild and heavy congestion, DT leads to a persistently full queue. Although the observed fluctuations are not significant, a full queue implies long delays, which may be unacceptable in many applications. PIE, in combination with all TCPs, achieves remarkably better results than CoDel. During heavy congestion, CoDel, similarly as in the DC scenario, oscillates between the maximum and target values, however at a significantly lower frequency. The queue size variability in the mildly congested network is depicted in Figure 6 and during heavy congestion in Figure 7.

Table 9. Packet drop ratio [%] in the Access Link scenario with common TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

		DT	PIE	CoDel
UC	New Reno	0.06	0.03	0.02
	Cubic	0.26	0.12	0.07
	C-TCP	0.01	0.01	0.01
MC	New Reno	0.75	0.85	0.71
	Cubic	0.89	1.31	1.15
	C-TCP	0.32	0.48	0.40
HC	New Reno	11.03	15.82	11.44
	Cubic	13.68	15.54	12.38
	C-TCP	10.92	15.89	11.68

TCP Cubic leads to the highest drop ratio compared to the other two TCPs (see Table 9). On the other hand, C-TCP, regardless of the queue management, provides the least packet losses. In the uncongested network, the TCP Cubic and DT combination achieves the worst results in terms of the drop ratio; replacing DT with any evaluated AQM considerably improves this metric. Nevertheless, the drop ratio is relatively small in this network load scenario.

In the mildly congested network, C-TCP once again provides the lowest and Cubic the highest drop ratio. During heavy congestion, it is visible that the drop ratio depends on the applied AQM, i.e., PIE causes slightly more losses than CoDel and DT; however, it provides the shortest and most stable queue at the same time.

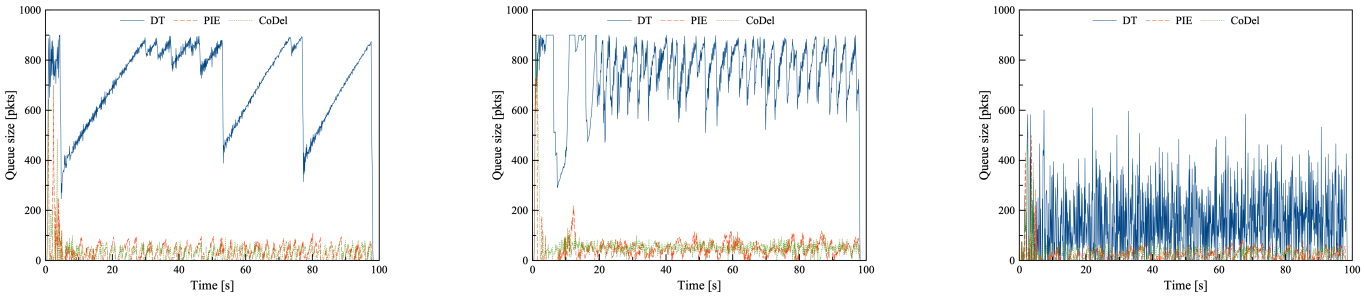
Similarly, as in the previous scenario, improvements can be proposed for the buffer management strategy. CoDel experiences similar problems with oscillations under heavy congestion, however a slower network causes less frequent oscillations. The average RTT is close to the nominal RTT, therefore it is clear that updating only the interval parameter according to the network speed, is not enough. It is necessary to increase the number of rejected packets per every μ. This number could depend on the increase rate of the average queue size.

Another problem that appears in this scenario is much lower fairness in uncongested and mildly congested networks, which results from varied RTTs. TCP flows with longer RTT occupy a much smaller part of bandwidth than TCP flows with shorter RTT. Deploying AQM already introduces randomization to which flow experiences loss. To improve this approach, both AQMs could adopt a simple mechanism to approximate the flow rate, e.g., borrowed from CHOKe [34]. CHOKe verifies if the upcoming packet belongs to the same flow as a randomly selected packet in the queue. If both packets belong to the same flow, it decides to drop the packet.

Summary:

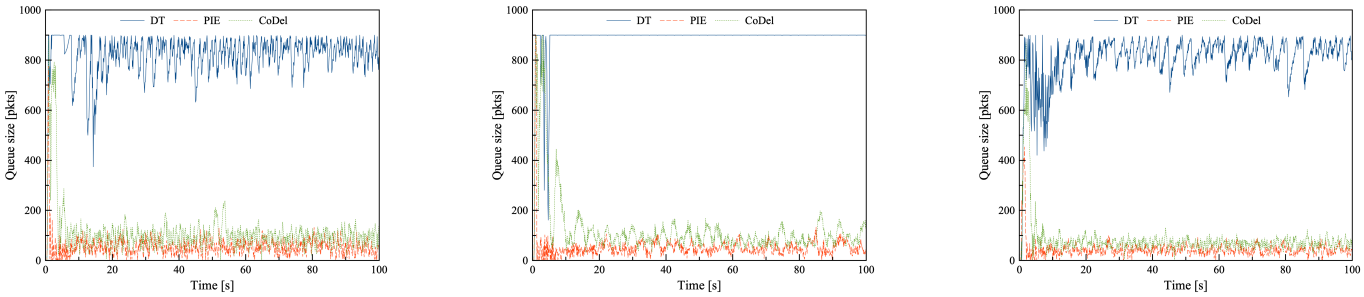
- Varied RTT significantly worsens the achievable fairness among competing flows for any combination of TCP congestion control mechanism and buffer management strategy.
- DT leads to a very high drop ratio, unfairness, and persistently full buffer. Applying PIE or CoDel improves the experienced delay and delay variations and, in most cases, fairness provided by the TCP congestion control mechanism.
- TCP Cubic benefits the most from replacing DT with AQM.
- In combination with DT, C-TCP provides a very high Jain's index, acceptable average queue size; however, high queue size variations and only in an uncongested network scenario.
- **The following combinations provide the best results in terms of satisfactory fairness, delay, and drop ratio at the same time: PIE with C-TCP (for uncongested**

network), PIE with Cubic (for mildly congested network), and PIE with any TCP (during heavy congestion).



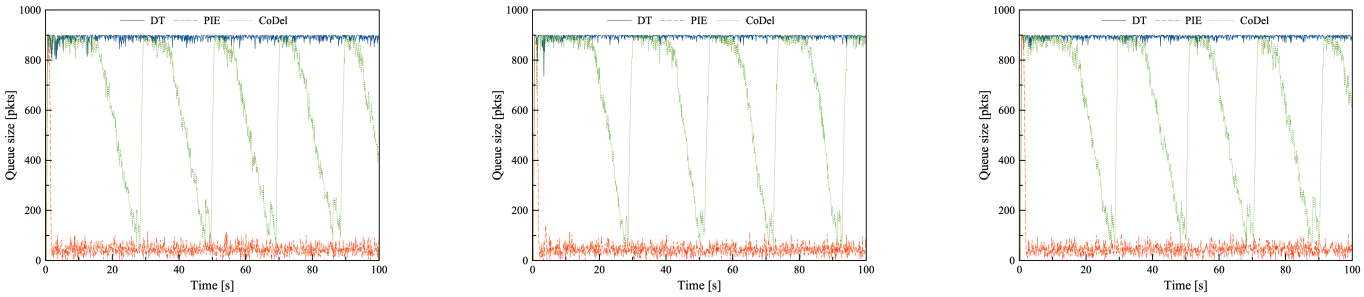
(**a**) New Reno (**b**) Cubic (**c**) C-TCP

Figure 5. The queue length of DT, PIE and CoDel in combination with (**a**) New Reno, (**b**) Cubic, and (**c**) C-TCP in the Access Link scenario with common TCPs in uncongested network.



(**a**) New Reno (**b**) Cubic (**c**) C-TCP

Figure 6. The queue length of DT, PIE and CoDel in combination with (**a**) New Reno, (**b**) Cubic, and (**c**) C-TCP in the Access Link scenario with common TCPs during mild congestion.



(**a**) New Reno (**b**) Cubic (**c**) C-TCP

Figure 7. The queue length of DT, PIE and CoDel in combination with (**a**) New Reno, (**b**) Cubic, and (**c**) C-TCP in the Access Link scenario with common TCPs during heavy congestion.

5.3. Data Center, Mixed TCPs

In the third scenario, the combination of all TCP versions is used simultaneously, i.e., every node implements a different congestion control mechanism. The results are shown in Tables 10 and 11. Table 10 shows the average throughput, bandwidth sharing, and fairness. The average throughput and standard deviation of the throughput (column AvgThr (SD)) are calculated for each node, thus for different TCP flavors, depending on the congestion load, it is the average of three, thirty, or three-hundred values. Column BD presents the percentage of the overall throughput achieved by the streams of a given TCP version. Column JI contains the Jain's index calculated either for flows using the same TCP

version (inter-fairness) or for all connections (intra-fairness). The results are presented for three different congestion levels and three evaluated buffer management strategies.

The most unfair bandwidth allocation is observed when DT is applied at the buffer. Even in the uncongested network, although the inter-fairness is preserved, the intra-fairness is rather poor, and it drops significantly as the congestion grows. In the uncongested and mildly congested network, New Reno acts most aggressively and utilizes more than 60% of the available bandwidth. On the other hand, C-TCP is the least competitive and suffers significant performance degradation. During heavy congestion, TCP Cubic overtakes the whole bandwidth.

Replacing DT with PIE or CoDel noticeably improves the inter- and intra-fairness in all scenarios, even during heavy congestion state. However, there are minor differences in terms of the average throughput; in most cases, TCP Cubic acts slightly more aggressively towards New Reno and C-TCP connections.

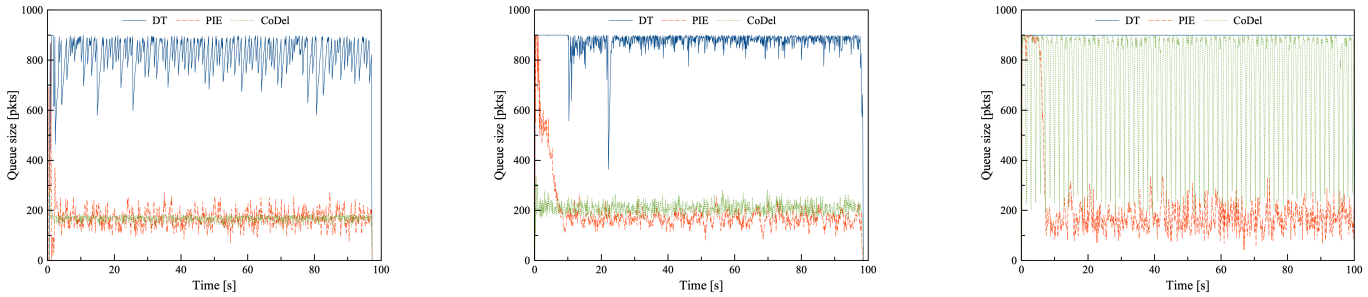
Table 10. The summary results in the data center scenario with mixed TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

		DT			**PIE**			**CoDel**		
		AvgThr (SD)	**BD [%]**	**JI**	**AvgThr (SD)**	**BD [%]**	**JI**	**AvgThr (SD)**	**BD [%]**	**JI**
UC	NReno	188 (37)	61	0.97	128 (6)	42	1.00	117 (9)	38	1.00
	Cubic	86 (5)	28	1.00	109 (8)	35	1.00	112 (5)	36	1.00
	C-TCP	34 (15)	11	0.89	71 (1)	23	1.00	78 (6)	25	1.00
	All	103 (71)	100	0.70	103 (26)	100	0.95	103 (19)	100	0.97
MC	NReno	21.44 (7,52)	69	0.89	9.85 (0.48)	31	1.00	9.84 (0.50)	31	1.00
	Cubic	4.55 (1.02)	15	0.95	10.97 (0.51)	35	1.00	10.98 (0.39)	35	1.00
	C-TCP	5.30 (1.95)	17	0.88	10.47 0.36	33	1.00	10.46 (0.37)	33	1.00
	All	10.43 (9)	100	0.57	10.43 (0.6)	100	1.00	10.43 (0.6)	100	1.00
HC	NReno	0 (0)	>1	0.15	0.91 (0.09)	29	0.99	0.95 (0.08)	30	0.99
	Cubic	3.17 (2.24)	∼99	0.67	1.21 (0.11)	38	0.99	1.16 (0.10)	37	0.99
	C-TCP	0 (0)	>1	0.01	1.06 (0.09)	33	0.99	1.06 (0.07)	33	1.00
	All	1.06 (2)	100	0.22	1.06 (0.15)	100	0.98	1.06 (0.12)	100	0.99

Table 11 shows the mean queue length and the standard deviation of queue length as well as the drop ratio for each scenario case. The queue length variation over time is presented in Figure 8. As far as queue behavior is concerned, similar dependencies can be observed as in the DC scenario with common TCP. CoDel provides a rather low and stable queue in the uncongested and mildly congested network. However, as the network load increases, CoDel's queue oscillates between the maximum and the target value, which causes high latency variability. The problem is discussed in the Data Center scenario with common TCPs (Section 5.1). PIE results in a comparable average queue size and queue size variation, regardless of the congestion level. During mild and heavy congestion, PIE causes 10% more packet losses than CoDel.

Table 11. Mean queue length (QS) and standard deviation (SD) of queue length [pkts] and packet drop ratio (DR) [%] in the Data center scenario with mixed TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

	DT		**PIE**		**CoDel**	
	QS (SD)	DR	QS (SD)	DR	QS (SD)	DR
UC	825 (57)	0.07	168 (34)	0.08	168 (11)	0.08
MC	881 (20)	0.56	167 (27)	1.67	209 (22)	1.49
HC	899 (1)	0.80	169 (52)	6.86	723 (217)	6.12



(**a**) UC (**b**) MC (**c**) HC

Figure 8. The queue length of DT, PIE and CoDel in combination with mix TCP flows in the Data Center scenario (**a**) in uncongested network, (**b**) during mild congestion, (**c**) during heavy congestion.

We summarize our findings are follows:

- Applying PIE or CoDel significantly improves the inter- and intra-fairness among competing flows using different TCP versions.
- CoDel maintains a short and stable queue only if the network is not overloaded. The average queue size provided by CoDel depends on the network load.
- PIE provides satisfactory and comparable results regarding fairness, drop ratio, and queue stability in all network load scenarios.

5.4. Access Link, Mixed TCPs

In the last scenario, the Access Link topology with varied RTTs is used along with all TCP versions, i.e., every node implements a different congestion control mechanism.

Table 12 gathers the average throughput, bandwidth sharing, and fairness. The average throughput and standard deviation of the throughput (column AvgThr (SD)) are calculated for each node, thus for different TCP flavors. Column BD presents the percentage of the overall throughput achieved by the streams of a given TCP version. Finally, column JI contains the Jain's index calculated either for flows using the same TCP version (inter-fairness) or for all connections (RTT-fairness). The results are presented for three different congestion levels and three evaluated buffer management strategies.

Preserving acceptable inter- and RTT-fairness, when all TCPs are used simultaneously, is much harder in Access Link scenario than in the Data Center scenario. In combination with DT, connections using TCP Compound can take only a disproportionately small percentage of the accessible resources. Thus, TCP New Reno and TCP Cubic consume almost all available bandwidth. This is because Cubic is more aggressive than the other two TCPs and the links dedicated to New Reno have the shortest average RTT values. TCP Cubic and C-TCP provide significantly better inter-fairness than New Reno. However, in all load scenarios, the overall RTT-fairness is rather poor.

Applying PIE or CoDel does not significantly affect the inter-fairness, actually C-TCP provides better results in combination with DT. Nonetheless, both AQMs improve the fair bandwidth distribution. From the conducted simulations, it cannot be clearly stated which version of TCP behaves the most aggressively or how RTT differentiation influences the achieved throughput. To fully understand and verify these dependencies, additional simulation studies would be required.

Table 12. The summary results in the Access Link scenario with mixed TCPs, UC—uncongested, MC—mildly congested, HC—eavily congested network.

		DT			PIE			CoDel		
		AvgThr (SD)	**BD [%]**	**JI**	**AvgThr (SD)**	**BD [%]**	**JI**	**AvgThr (SD)**	**BD [%]**	**JI**
UC	NReno	16.24 (16.28)	52	0.60	11.84 (11.48)	38	0.61	10.93 (10.27)	35	0.63
	Cubic	14.07 (5.99)	45	0.89	13.81 (3.60)	44	0.96	15.51 (2.66)	50	0.98
	C-TCP	0.90 (0.33)	3	0.92	5.58 (2.96)	18	0.84	4.77 (3.63)	15	0.72
	All	10.40 (11.26)	100	0.49	10.41 (7.23)	100	0.70	10.40 (7.30)	100	0.70
MC	NReno	1.17 (0.95)	37	0.61	1.47 (1.17)	46	0.62	1.42 (1.04)	45	0.66
	Cubic	1.67 (0.52)	53	0.91	1.01 (0.21)	32	0.96	1.07 (0.21)	34	0.96
	C-TCP	0.33 (0.10)	10	0.92	0.70 (0.41)	22	0.75	0.68 (0.36)	21	0.79
	All	1.06 (0.83)	100	0.62	1.06 (0.79)	100	0.65	1.06 (0.71)	100	0.69
HC	NReno	0.17 (0.10)	54	0.76	0.12 (0.05)	38	0.86	0.12 (0.05)	37	0.86
	Cubic	0.09 (0.03)	30	0.89	0.09 (0.02)	28	0.94	0.10 (0.02)	33	0.95
	C-TCP	0.05 (0.01)	16	0.97	0.11 (0.04)	33	0.90	0.09 (0.02)	30	0.94
	All	0.11 (0.08)	100	0.65	0.11 (0.04)	100	0.88	0.11 (0.04)	100	0.90

Table 13 shows the mean queue length and the standard deviation of queue length, as well as the drop ratio for each scenario case. The queue length variation over time is presented in Figure 9.

The observed queue behavior is similar to previous scenarios. DT allows filling the queue, which results in a large average queue size and, consequently, long delays. In the uncongested network, the queue oscillations, therefore delay variations, are the highest (see Figure 9a). In the uncongested and mildly congested network, PIE and CoDel provide similar results, i.e., rather short and stable queues. CoDel, as depicted in Figure 9c, fluctuates between the maximum and the target value of queue size in the heavily congested network, which results in unacceptably high queue size variations. PIE acts similarly in all load scenarios and provides comparable average queue size and rather small variations. However, during heavy congestion, it causes considerably more packet losses than the other two approaches.

Table 13. Mean queue length (QS) and standard deviation (SD) of queue length [pkts] and packet drop ratio (DR) [%] in the Access Link scenario with mixed TCPs, UC—uncongested, MC—mildly congested, HC—heavily congested network.

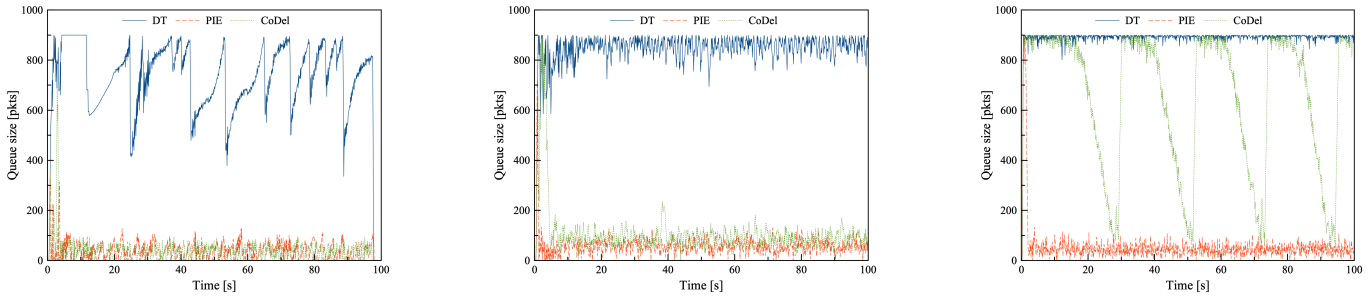
	DT		PIE		CoDel	
	QS (SD)	**DR**	**QS (SD)**	**DR**	**QS (SD)**	**DR**
UC	741 (114)	0.11	38 (29)	0.03	35 (24)	0.03
MC	856 (33)	0.87	58 (24)	0.91	89 (32)	0.82
HC	893 (8)	12.11	44 (21)	15.84	632 (281)	11.98

A similar problem with fairness in the uncongested and moderately congested networks is observed in the Access Link scenario with common TCPs. Once again, the RTT fairness could be improved by deploying the dropping function, which would more accurately penalize the faster flows.

A summary of our findings is given below.

- Varied RTT significantly worsens the results in terms of achievable RTT- and inter-fairness.
- Applying PIE or CoDel improves the RTT-fairness among competing flows using different TCP versions, however, unsatisfactorily.

- C-TCP, combined with DT, provides very high inter-fairness even in the presence of connections using more aggressive TCP variants.
- PIE is the only buffer management strategy that provides acceptable results in terms of fairness, and queue stability in all load scenarios, however, at the cost of a higher drop ratio.



(**a**) UC (**b**) MC (**c**) HC

Figure 9. The queue length of DT, PIE and CoDel in combination with mix TCP flows in the Access Link scenario (**a**) in uncongested network, (**b**) during mild congestion, (**c**) during heavy congestion.

6. Conclusions

In this paper, an extensive evaluation of cross-layer interactions between commonly used TCP flavors, including New Reno, Cubic, Compound, and recent Active Queue Management strategies, was conducted to answer fundamental questions:

1. To what extent an AQM influences the end-to-end congestion control mechanism?
2. What are the effects of network parameters, such as RTT variation and congestion level, on the combined TCP and AQM performance?
3. Can cross-layer interaction between TCP and AQM overcome the challenges of network heterogeneity and variability?
4. Is there one universal TCP-AQM combination for all cases?

Re (1) The simulation results revealed that in combination with DT, none of the evaluated TCP provides fair bandwidth sharing, especially when the network load increases. Moreover, when all TCPs are used simultaneously without any AQM TCP Compound, which is a delay-based algorithm, there is a decrease in performance when competing with TCP flows from the loss-based group. On the other hand, TCP Cubic is the most aggressive congestion control mechanism, utilizing over 50% of available bandwidth. Applying PIE or CoDel significantly improves the inter- and intra-fairness provided by the evaluated congestion control mechanisms.

Re (2) Varied RTT significantly worsens the performance of all TCP-AQM combinations in terms of achievable fairness among competing flows. Moreover, with the growing network load, the fairness decreases even more, the drop ratio is higher, and providing a low and stable queue becomes very challenging.

Re (3) DT always leads to a persistently full buffer, which results in long delays and high delay variations; furthermore, the higher network load, the fewer flows can transmit data packets efficiently. Applying any AQM noticeably improves the RTT-fairness; though, the aggressiveness of the congestion control protocol cannot be eliminated. In most cases, PIE and CoDel allow maintaining relatively short and stable queues, which reduces the experienced delay and delay variations, and improves the inter- and intra-fairness among competing flows.

Re (4) No combination would result in the highest fairness, low drop ratio, and low and stable queue simultaneously, regardless of the congestion level or RTT variability. In general, CoDel, especially in combination with C-TCP, maintains the shortest queues and provides satisfactory fairness, however only in an uncongested network. PIE is the only

AQM that, with all TCPs, in all load scenarios provides similar results in terms of queue length and stability, reasonable fairness, however at the cost of a slightly higher drop ratio.

PIE and CoDel configuration parameters were supposed to be auto-tuned and insensitive to varied round-trip delays, link rates, and traffic loads. Unfortunately, there are some operational ranges for which both AQMs work as desired and provide low and stable queuing delay. Outside those ranges some additional configuration is beneficial. Especially, the target delay should be reduced, according to the network speed, for both AQMs.

PIE and TCP Cubic seem to represent the most universal combination that works well in both the high-speed network with low delays and the access link connecting an institution to an ISP. PIE is insensitive to traffic load, and TCP Cubic deals better with varied RTT in terms of fairness provided. CoDel with C-TCP works well in the network with similar RTTs and low traffic load. C-TCP is a good choice in an uncongested network when we cannot deploy the AQM, because it provides the highest fairness and prohibits filling the entire buffer.

The results obtained in this paper are contradictory to the results presented in [16]. However, it is not surprising, taking into account the configuration of the simulation environment. We believe that the simulation parameters and analyses submitted in this paper apply to current technological standards and represent more common cases. Moreover, the results confirm the limitations of both algorithms suggested in [11] for high-speed network and heavily congested networks.

While there is no universal solution for all scenarios, it is clear that replacing DT with PIE or CoDel should be a must, especially when we consider that aggressive TCP Cubic is likely to be the most common solution in the future. As part of future work, we will repeat the simulations using real TCP implementations.

Funding: This work was carried out within the statutory research project of the Department of Computer Networks and Systems.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Postel, J. Service mappings. RFC 795, RFC Editor. Available online: https://www.rfc-editor.org/info/rfc795 (accessed on 22 August 2021).
2. Afanasyev, A.; Tilley, N.; Reiher, P.; Kleinrock, L. Host-to-Host Congestion Control for TCP. *IEEE Commun. Surv. Tutor.* **2010**, *12*, 304–342. [CrossRef]
3. Baker, F.; Fairhurst, G. IETF Recommendations Regarding Active Queue Management. Available online: https://www.rfc-editor.org/info/rfc7567 (accessed on 22 August 2021).
4. Miras, D.; Bateman, M.; Bhatti, S. 2008 22nd IEEE International Conference on Advanced Information Networking and Applications. Available online: http://toc.proceedings.com/05076webtoc.pdf (accessed on 22 August 2021).
5. Alrshah, M.A.; Othman, M.; Ali, B.; Mohd Hanapi, Z. Comparative study of high-speed Linux TCP variants over high-BDP networks. *J. Netw. Comput. Appl.* **2014**, *43*, 66–75. [CrossRef]
6. Turkovic, B.; Kuipers, F.A.; Uhlig, S. Fifty Shades of Congestion Control: A Performance and Interactions Evaluation. *arXiv* **2019**, arXiv:1903.03852.
7. Lin, J.; Cui, L.; Zhang, Y.; Tso, F.P.; Guan, Q. Extensive evaluation on the performance and behaviour of TCP congestion control protocols under varied network scenarios. *Comput. Netw.* **2019**, *163*, 106872, doi:10.1016/j.comnet.2019.106872. [CrossRef]
8. Callegari, C.; Giordano, S.; Pagano, M.; Pepe, T. Behavior analysis of TCP Linux variants. *Comput. Netw.* **2012**, *56*, 462–476. [CrossRef]
9. Poojary, S.; Sharma, V. Analysis of multiple flows using different high speed TCP protocols on a general network. *Perform. Eval.* **2016**, *104*, 42–62. [CrossRef]
10. Abdeljaouad, I.; Rachidi, H.; Fernandes, S.; Karmouch, A. Performance analysis of modern TCP variants: A comparison of Cubic, Compound and New Reno. In Proceedings of the 2010 25th Biennial Symposium on Communications, Kingston, ON, Canada, 12–14 May 2010; pp. 80–83.
11. Kuhn, N.; Ros, D.; Bagayoko, A.B.; Kulatunga, C.; Fairhurst, G.; Khademi, N. Operating ranges, tunability and performance of CoDel and PIE. *Comput. Commun.* **2017**, *103*, 74–82. [CrossRef]

12. Schwardmann, J.; Wagner, D.; Kühlewind, M. Evaluation of ARED, CoDel and PIE. In *Advances in Communication Networking*; Kermarrec, Y., Ed.; Springer International Publishing: Cham, Switzerland, 2014; pp. 185–191.
13. Dash, P.K.; Bisoy, S.K. Analysis of AQM router of network supporting multiple TCP flows. In Proceedings of the 2014 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 18–20 December 2014; pp. 1–5.
14. Chydzinski, A.; Brachman, A. Performance of AQM Routers in the Presence of New TCP Variants. In Proceedings of the 2010 Second International Conference on Advances in Future Internet, Venice, Italy, 18–25 July 2010; pp. 88–93.
15. Vyakaranal, S.B.; Naragund, J.G. Performance Evaluation of TCP using AQM Schemes for Congestion Control. In Proceedings of the 2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC), Bangalore, India, 9–10 February 2018; pp. 1–6.
16. Grazia, C.A.; Patriciello, N.; Klapez, M.; Casoni, M. Transmission Control Protocol and Active Queue Management together against congestion: cross-comparison through simulations. *SIMULATION* **2019**, *95*, 979–993. [CrossRef]
17. Andrew, L.; Marcondes, C.; Floyd, S.; Dunn, L.; Guillier, R.; Gang, W.; Eggert, L.; Ha, S.; Rhee, I. Towards a Common TCP Evaluation Suite. Available online: http://www.icsi.berkeley.edu/pubs/networking/pfldnet2008-submitted.pdf (accessed on 22 August 2021).
18. Gurtov, A.; Henderson, T.; Floyd, S.; Nishida, Y. The NewReno Modification to TCP's Fast Recovery Algorithm. Available online: https://www.rfc-editor.org/info/rfc6582 (accessed on 22 August 2021).
19. Jacobson, V. Congestion Avoidance and Control. *SIGCOMM Comput. Commun. Rev.* **1988**, *18*, 314–329. [CrossRef]
20. Floyd, S. HighSpeed TCP for Large Congestion Windows. Available online: https://www.rfc-editor.org/info/rfc3649 (accessed on 22 August 2021).
21. Ha, S.; Rhee, I.; Xu, L. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *SIGOPS Oper. Syst. Rev.* **2008**, *42*, 64–74. [CrossRef]
22. Xu, L.; Harfoush, K.; Rhee, I. Binary increase congestion control (BIC) for fast long-distance networks. In Proceedings of the IEEE INFOCOM 2004, Hong Kong, China, 7–11 March 2004; Volume 4, pp. 2514–2524.
23. Tan, K.; Song, J.; Zhang, Q.; Sridharan, M. A Compound TCP Approach for High-Speed and Long Distance Networks. In Proceedings of the IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, Barcelona, Spain, 23–29 April 2006; pp. 1–12.
24. Brakmo, L.S.; Peterson, L.L. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE J. Sel. Areas Commun.* **1995**, *13*, 1465–1480. [CrossRef]
25. Nichols, K.; Jacobson, V. Controlling Queue Delay. *Queue* **2012**, *10*, 20–34. [CrossRef]
26. Adams, R. Active Queue Management: A Survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1425–1476. [CrossRef]
27. Pan, R.; Natarajan, P.; Baker, F.; White, G. Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem. Available online: https://www.rfc-editor.org/info/rfc8033 (accessed on 22 August 2021).
28. Imputato, P.; Avallone, S.; Tahiliani, M.P.; Ramakrishnan, G. Revisiting design choices in queue disciplines: The PIE case. *Comput. Netw.* **2020**, *171*, 107136. [CrossRef]
29. Pan, R.; Natarajan, P.; Piglione, C.; Prabhu, M.S.; Bernstein, A.; Baker, F.J. Adapting Proportional Integral Controller Enhanced Algorithm for Varying Network Conditions in a Network Environment. U.S. Patent 9,674,104, 15 January 201 .
30. Nichols, K.; Jacobson, V.; McGregor, A.; Iyengar, J. Controlled Delay Active Queue Management. Available online: https://www.rfc-editor.org/info/rfc8289 (accessed on 22 August 2021).
31. Hoeiland-Joergensen, T.; McKenney, P.; Taht, D.; Gettys, J.; Dumazet, E. The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm. Available online: https://www.rfc-editor.org/info/rfc8290 (accessed on 22 August 2021).
32. Levasseur, B.; Claypool, M.; Kinicki, R. A TCP CUBIC Implementation in Ns-3. In *WNS3 '14: Proceedings of the 2014 Workshop on ns-3*; Association for Computing Machinery: New York, NY, USA, 2014; [CrossRef]
33. Craig, K. Available online: https://web.cs.wpi.edu/~rek/Adv_Nets/Fall2014/TCP_Compound_Project.pdf (accessed on 22 August 2021).
34. Pan, R.; Prabhakar, B.; Psounis, K. CHOKe—A stateless active queue management scheme for approximating fair bandwidth allocation. In Proceedings of the IEEE INFOCOM 2000, Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 26–30 March 2000; Volume 2, pp. 942–951. [CrossRef]