2016 IEEE 41st Conference on Local Computer Networks

# Toward Coexistence of Different Congestion Control Mechanisms

Mario Hock, Roland Bless, Martina Zitterbart
Karlsruhe Institute of Technology
Karlsruhe, Germany
E-Mail: {mario.hock, bless, zitterbart}@kit.edu

***Abstract*—Interactive and delay-sensitive applications constitute an important and growing part of the Internet. Today, low delays can only be achieved if there are no congested links along the path. At a bottleneck, the commonly used congestion control mechanisms induce high queuing delay and packet loss. Specialized congestion control mechanisms achieving a low queuing delay in an otherwise congested network exist. However, if their flows share a bottleneck with traditionally congestion controlled flows, they get suppressed or loose their low-delay property. This paper presents three approaches – Separate Paths, Separate Queues and Limited Queue – that enable coexistence of congestion control mechanisms that are optimized for different goals, e.g., interactive applications and bulk transfers. The measurement results show that concurrent operation of different congestion control mechanisms can be successfully implemented this way.**

***Index Terms*—Congestion control, Low delay, Software-defined Networking**

## I. Introduction

The Internet serves an ever increasing variety of applications. Their requirements range from bulk-transfers (i.e., high volume data transfers) with high throughput requirements to interactive or real-time applications that need low and bounded delays. Some of these applications, like high-quality video conferences or Ultra-HD Virtual Reality, require both: high throughput as well as *low delay*. The latter is crucial because of the interactive characteristics of the application, which typically work best if the end-to-end delay is below 150 ms.

If there is a bottleneck on a network path, congestion control that is used by the senders has crucial impact on queuing delay and, therefore, on end-to-end delay. Established congestion control mechanisms (e.g., CUBIC TCP, Compound TCP) tend to fill up all available buffer space until packet-loss occurs, thereby creating large *standing queues* [1]. In the following we will denote such congestion control mechanisms and corresponding flows as *"queue-filling"*. One approach to reduce standing queues is to use *Active Queue Management* (AQM), which aims at keeping the average buffer occupancy low, thereby reducing the queuing delay [2]. Another approach is to use congestion control mechanisms that aim at achieving a low end-to-end delay (e.g., TCP Vegas [3], CDG [4] and YeAH TCP [5]). Instead of reacting on packet loss, they try to estimate the buffer occupancy level (or its variation) and back off as soon as they assume congestion. However, their disadvantage is that they cannot reasonably coexist with today's congestion control mechanisms, as explained in the following.

This work has been supported in the bwNET100G+ project by the Ministry of Science, Research and the Arts Baden-Württemberg (MWK). The authors alone are responsible for the content of this paper.

Flows with a low-delay congestion control (in short: *low-delay flows*) try to avoid packet buffering as far as possible in order to reduce the induced queuing delay. A *queue-filling flow*, in contrast, will not limit its buffer consumption. It regularly increases its congestion window until packet loss occurs, thereby occupying as much buffer capacity as available. This means that the bottleneck buffer will usually contain significantly more packets from queue-filling flows than from low-delay flows. But the share of the transmission capacity a flow gets under congestion is correlated to the share of the buffer capacity the flow occupies. Thus, low-delay flows get only a small amount of the bottleneck bandwidth.

YeAH TCP and CDG have a compatibility mode to avoid being completely suppressed by queue-filling flows. When such a situation is detected, they switch to a TCP Reno-like behavior. But this also means that they sacrifice their low-delay goals, in these cases. Either way, the benefit from using a low-delay congestion control is nullified in the presence of queue-filling flows. This constitutes a large obstacle for a gradual deployment of low-delay congestion control in the Internet.

## II. CoEx-Approaches

In this section we present three approaches that can facilitate the coexistence of low-delay and queue-filling flows (short *CoEx-Approaches*): *"Separate Paths"*, *"Separate Queues"* and *"Limited Queue"*. They are explained in the following along with ways of implementing them with off-the-shelve OpenFlow hardware. All CoEx-Approaches work on *aggregates*. One aggregate consists of all low-delay flows, another aggregate consists of all queue-filling flows.

*1) Separate Paths:* All low-delay flows can be routed over a separate path so that they do not share any link with the queue-filling flows. Consequently, the queuing delay in one of the paths does not affect the queuing delay of the other path. Although using separate paths is not easily possible with traditional IP-routing, it can be implemented with OpenFlow, *MPLS* [6], or *segment routing* [7].

With OpenFlow, forwarding rules are programmed in form of *flow-rules* that match on the packet header and define how a

DOI 10.1109/LCN.2016.94

matching packet will be processed. In contrast to traditional IP-routing, this matching is not limited to *longest prefix matching*. This means that the use of separate paths is naturally supported by OpenFlow. This approach can, thus, be realized on any OpenFlow switch, provided that redundant links and paths are available in the network.

*2) Separate Queues:* Using a separate queue for the low-delay flows has two effects. First, it decouples the transmission rate from the buffer utilization and leaves the bandwidth distribution to a *scheduler*. Second, the queuing delay in one of the queues does not affect the queuing delay of other queues. The use of separate queues for different traffic classes is a well-known quality-of-service mechanism (e.g., DiffServ, OpenFlow).

With OpenFlow different queues can be employed by the *"setQueue-action"*, which was introduced in version 1.3 of the OpenFlow specification. The queues themselves are not programmed by OpenFlow. For their setup another protocol, like *OFconfig*, is needed. The setQueue-action, however, is marked as *optional* in the OpenFlow specification and, thus, is not supported by every OpenFlow switch.

*3) Limited Queue:* If queue-filling flows and low-delay flows share a common queue, the queue occupancy of the queue-filling flows has to be limited. Otherwise there will be a high queuing delay for all flows. This will also reduce the suppression of low-delay flows, in terms of throughput. However, care has to be taken to do not unnecessarily drop packets of the low-delay flows.

There are usually no dedicated mechanisms that explicitly limit the buffer occupancy of particular flows in typical network hardware. However, the desired effect can still be achieved indirectly with the use of *rate limiters*. Limiting the aggregate of queue-filling flows to a value well below 100% of the capacity of the output port (e.g., 50%), means that the queue-filling flows alone cannot build up a queue, thus, the resulting queuing delay can be controlled by the low-delay flows.

In OpenFlow, rate limiters are available since version 1.3 of the OpenFlow specification as an optional feature and are called *OpenFlow meters*. They can be applied to a number of flow-rules and monitor the combined throughput of the flows. If the limit is exceeded, packets are dropped.

## III. Evaluation

For the evaluation of the proposed approaches we set up a physical testbed that contains a *wide area network (WAN)*. The testbed set-up is depicted in Fig. 1. It has two configurations, one with redundant links at the bottleneck switch to examine the *Separate Paths approach* (see Fig. 1a) and one without redundant links to examine the *Separate Queues approach* and the *Limited Queue approach* (see Fig. 1b).

The WAN is a 10 Gbit/s slice of a 100 Gbit/s research network that connects multiple universities. The network is sliced below *Layer 2* and our slice was exclusively used for the experiments, thus, there are no side-effects from other slices. The base RTT on this path is about 6.2 ms. Traffic was generated with *iperf*[1]. The measurement data (throughput, CWnd, RTT) was collected with the tool *CPUnetLOG*[2] and the Linux kernel module *tcp_probe*. Since the bottleneck switch can only forward 1 Gbit/s (Fig. 1b), resp. 2 Gbit/s (Fig. 1a), the WAN is never the bottleneck and, thus, no CoEx-Approach is required in the WAN. The buffer sizes at the bottleneck switch are set to 625 kByte per queue.

In all experiments, TCP Reno was used to generate the queue-filling flows. For low-delay flows a modified version of TCP Vegas that we call *TCP Vegas*$_{20/40}$ was used. The regular TCP Vegas was not able to saturate a (congestion-free) 1 Gbit/s link. For the modified version, the number of packets that TCP Vegas strives to always hold in the bottleneck buffer was increased from 2 – 4 packets to 20 – 40 packets. On a 1 Gbits/s link, this results in the same queuing delay as the original values on a 100 Mbit/s link, on which the regular TCP Vegas works fine.

In all experiments sender `S1` starts first, sender `S2` starts 60 seconds later; all connections last for 300 seconds. The following plots show the time interval between second `50` and second `120`, since this interval contains all relevant phases and transitions.

The buffer utilization at the bottleneck can be determined from the RTT plots. With empty buffers, the RTT is at its minimum (e.g., during the first minute when only one sender is active). Non-empty buffers add queuing delay, which lead to higher RTT values depending on the buffer utilization and the drain rate. Note that in the *Separate Queues approach* the drain rate per queue is lower (500 Mbit/s, if both queues are not empty), thus, the maximal queuing delay is significantly higher in these experiments.

### A. Separate Paths

In this measurement experiment both senders, `S1` and `S2`, transmit a queue-filling flow as well as a low-delay flow that are forwarded over separate paths.

As expected, the queue-filling flows do not influence the low-delay flows with respect to throughput (Fig. 2a and 2b) and delay (Fig. 2c and 2d). The bottleneck links on both paths are fully utilized. This means that both aggregates (low-delay flows and queue-filling flows) can exploit the same bandwidth.

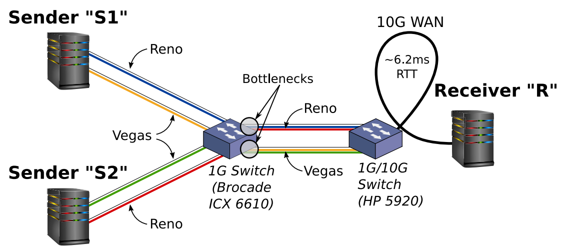
The buffer at the bottleneck for the queue-filling flows is repeatedly filled up to exhaustion which causes a high and strongly fluctuating RTT and frequent packet drops (see Fig. 2c). But this does not affect the low-delay flows. As shown in Fig. 2d, a low and steady RTT is maintained and no packet loss could be observed. In contrast, the low-delay flows only show a slight temporary increase of the RTT after `S2` started. Thereafter, both low-delay flows experience a rather stable RTT with very low fluctuations.

### B. Separate Queues

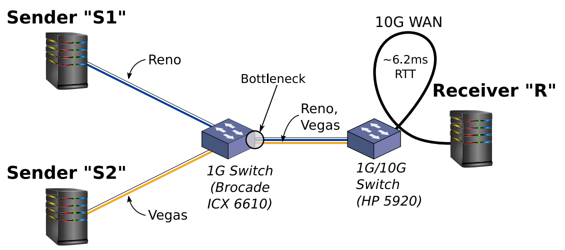
In this experiment, only one link is used per sender and at the bottleneck (see Fig. 1b). `S1` transmits a single queue-filling

[1]https://iperf.fr/

[2]https://www.tm.kit.edu/CPUnetLOG

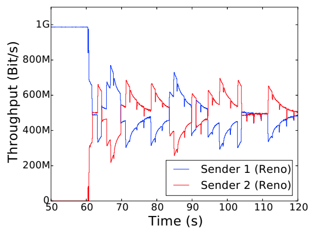


(a) Redundant links: *Separate Paths approach*

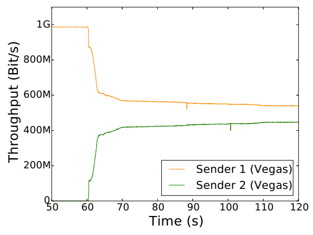


(b) No redundant links: *Separate Queues approach* and *Limited Queue approach*

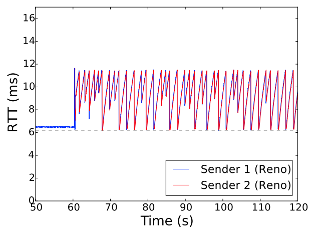
Fig. 1. Testbed Set-up



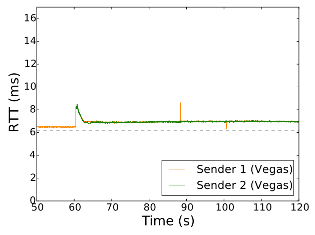
(a) TCP Reno *Throughput*



(b) TCP Vegas$_{20/40}$ *Throughput*

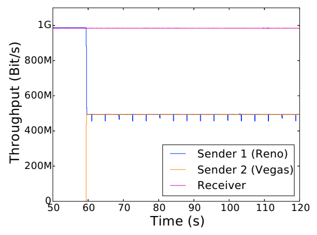


(c) TCP Reno *RTT*

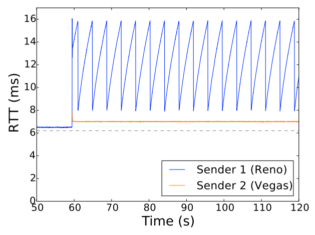


(d) TCP Vegas$_{20/40}$ *RTT*

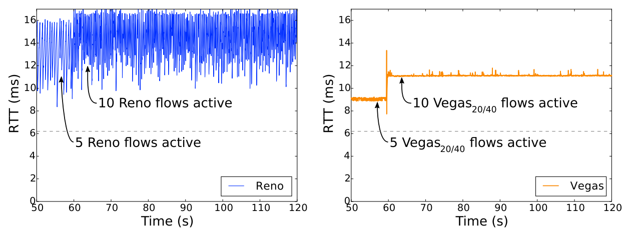
Fig. 2. *Separate Paths approach*



(a) Throughput



(b) RTT

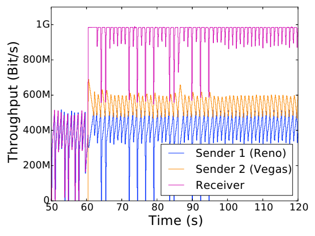


(c) Multiple flows (RTT)

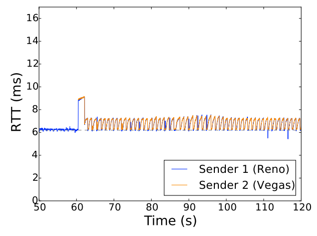
Fig. 3. *Separate Queues approach*

flow, S2 transmits a single low-delay flow to R. Both flows share a single output port at the switch. But still they are enqueued into separate queues (as scheduling policy *weighted round robin (WRR)* with equal weights is used). This way, an equal distribution of the available bandwidth among the two flows is achieved (see Fig. 3a), even though the queue sizes differ significantly (cf. different RTTs in Fig. 3b). While the queue-filling flow periodically fills the queue up to exhaustion and causes packet loss, the low-delay flow is able to keep a low queuing delay and experiences no packet loss. A full utilization of the bottleneck link is achieved, not only after second 60 when both flows are active, but also before, when only one flow is active. This demonstrates a particular advantage of the *Separate Queues approach*. If one of the flow aggregates does not use its share of the bandwidth, freely available capacity is automatically utilized by the other aggregate.

We also conducted experiments with multiple flows (see Fig. 3c). In the beginning, 5 low-delay and 5 queue-filling flows are active. After second 60, the number of flows is increased to 10 flows each. The results show that the behavior with multiple flows is similar to the previous experiment. A noticeable difference for the low-delay flows is that the RTT linearly depends on their number. Still the RTT shows only minor fluctuations.



(a) Throughput



(b) RTT

Fig. 4. *Limited Queue approach*

## C. Limited Queue

For this experiment the same set-up is used as in the previous one, but instead of separate queues a rate limiter is used that limits the queue-filling flow to 500 Mbit/s (with an accepted *burst-size* of 1 Mbit). In general, the rate limit applies to the aggregate of queue-filling flows.

Due to the rate limit, the queue-filling flow alone cannot fill the buffer. Together with the low-delay flow the buffer can be filled, as can be seen in Fig. 4b, especially around second 60. It has to be noted that, because of the rate-limit of 50%, the queue-filling flow can queue the same amount of data as the low-delay flow. If it queues more data, its transmission rate increases over 50% and the rate limiter is triggered. Most of the time, a small queue is achieved for both flows. However, the here used TCP Reno (but also other queue-filling congestion controls) are known to be not able to keep a constantly high throughput with such short queues. Indeed, the mean throughput of the queue-filling flow is significantly lower than the rate limit of 500 Mbit/s, throughout the experiment (see Fig. 4a). The underutilization is detected by the low-delay flow, which is able to use the free capacity to some extent. Therefore, it gets a higher average throughput.

The *Limited Queue approach* enables the low-delay congestion control to determine an acceptable queuing delay. With a rate limit of 50%, the queue-filling flows can contribute to the same extent to the queuing delay as the low-delay flows (analog for other rate limits). We call this *delay-fairness*. The bandwidth, however, is not equally shared, since the queue-filling congestion control cannot use the buffer as efficiently as the low-delay congestion control.

## IV. Related Work

The coexistence of loss-based and low-delay congestion control is also discussed in [8]. It describes a *Dual Queue Coupled AQM* that allows the coexistence of TCP Reno/Cubic TCP with a modified version of *Datacenter-TCP*. This AQM uses separate but interlinked queues that provide flow-rate fairness across the queues. The *Dual Queue Coupled AQM* is not available in any off-the-shelf router/switch, yet. In [9] the coexistence of low-delay and loss-based congestion control is evaluated on a tail-drop and an AQM-enabled bottleneck. The results show that the multimedia congestion control SCC and the low-delay congestion control CDG coexist well, even though they were designed independently from each other. Furthermore, it was shown that the Active Queue Mechanism *PIE* can be used to significantly increase the quality of a video-stream (transmitted with SCC) that shares a bottleneck with a Cubic TCP connection. In [10] it is shown that for TCP Reno no single AQM configuration is ideal for different kinds of traffic. Therefore, the feasibility of a programmable data plane is investigated. Based on [10], [11] suggests to use fair queuing to isolate each flow into its own subqueue and use SDN to equip each subqueue with the AQM that is most appropriate for the respective application. Our CoEx-Approaches differ from router-based congestion control, like XCP [12], since in our approaches the network does not allocate any resources to individual end-system but only ensures the coexistence of the different congestion controls. Furthermore, our work leverages off-the-shelf OpenFlow enabled hardware that is increasingly deployed in today's networks.

## V. Conclusion

In this paper we investigated different approaches that facilitate the concurrent operation of different congestion control mechanisms, with focus on low-delay congestion control. Low delay is important for modern interactive applications, but a gradual deployment of suitable congestion control mechanisms is not easily possible in the current Internet. The presented CoEx-Approaches (Separate Paths, Separate Queues and Limited Queue) can be realized with off-the-shelf hardware; in this paper we focused on OpenFlow switches.

With the *Separate Paths* and *Separate Queues approach* the properties of the different congestion control mechanisms are mostly unchanged. A fair coexistence is enabled since the approaches *decouple* the buffer utilization of a flow aggregate from the bandwidth share it gets. Hereby, the *Separate Queues approach* has the advantage that capacity that is not used by one of the aggregates can be automatically used by the other aggregate. Also, a more flexible allocation of bandwidth is possible and no redundant paths are required in the network. The *Limited Queue approach* enables a concept that we call *delay-fairness*. It empowers the low-delay congestion control to determine an acceptable queuing delay and forces the otherwise *queue-filling* congestion control variants to respect this limit.

In the future we focus on the improvement of congestion control mechanisms, in order to support low-delay and high throughput in high-speed environments and on the signalization of traffic characteristics and requirements, esp. in SDN networks. Moreover, we plan to investigate AQMs and fair queuing schedulers as further approaches to facilitate the coexistence of different congestion control mechanisms.

## References

[1] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *ACM Queue*, vol. 9, no. 11, pp. 40–54, Nov. 2011.

[2] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567, IETF, Jul. 2015.

[3] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *SIGCOMM '94*. New York, NY, USA: ACM, 1994, pp. 24–35.

[4] D. A. Hayes and G. Armitage, "Revisiting TCP Congestion Control Using Delay Gradients," in *NETWORKING'11*. Springer-Verlag, 2011, pp. 328–341.

[5] A. Baiocchi, A. P. Castellani, and F. Vacirca, "YeAH-TCP: Yet Another Highspeed TCP," in *Int. Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT)*, vol. 7, 2007, pp. 37–42.

[6] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, IETF, Jan. 2001.

[7] C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," IETF, Internet-Draft draft-ietf-spring-segment-routing-09, Jul. 2016, Work in Progress.

[8] B. Briscoe, O. Bondarenko, K. D. Schepper, and I. J. Tsang, "DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput," IETF, Internet-Draft draft-briscoe-aqm-dualq-coupled-01, Mar. 2016, Work in Progress.

[9] N. Iya, N. Kuhn, F. Verdicchio, and G. Fairhurst, "Analyzing the Impact of Bufferbloat on Latency-Sensitive Applications," in *IEEE International Conference on Communications (ICC)*, June 2015, pp. 6098–6103.

[10] A. Sivaraman, K. Winstein, S. Subramanian, and H. Balakrishnan, "No Silver Bullet: Extending SDN to the Data Plane," in *HotNets-XII*. ACM, 2013, pp. 19:1–19:7.

[11] S. Varma, *Internet Congestion Control*. Morgan Kaufmann, 2015.

[12] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-delay Product Networks," in *SIGCOMM '02*. ACM, 2002, pp. 89–102.