Decoupling Delay and Resource Sharing Targets with Efficient Core-Stateless AQM

Ferenc Fejes, Sándor Laki, Gergő Gombos
ELTE Eötvös Loránd University
Budapest, Hungary
fejes@inf.elte.hu

Szilveszter Nádas, Zoltán Kiss
Ericsson
Budapest, Hungary
szilveszter.nadas@ericsson.com

ABSTRACT

Core-Stateless resource sharing control identifies flow aggregates in the edge and marks packets based on the relevant policies. Within the core of the network, no flow identification or policy knowledge is needed to realize the desired resource sharing. As packet marking needs trust and the behavior of core-routers is independent of the number of flow aggregates, it is a natural choice for high-speed closed domain networks. In this demo paper, the Per Packet Value (PPV) core-stateless resource sharing framework is extended with an efficient AQM algorithm that decouples delay and resource sharing targets (separating the importance of a packet and its delay requirement). The marking algorithms of PPV and the proposed Core-Stateless AQM method implemented in DPDK are evaluated under various traffic loads in a network with a 10 Gbps bottleneck. Finally, we demonstrate how the proposed solution protects flows (e.g. download, gaming) of different delay requirements and resource sharing policies from a misbehaving low-delay and high-priority flow.

CCS CONCEPTS

• **Networks** → **Packet scheduling**; **Network resources allocation**.

KEYWORDS

Resource Sharing, QoS, Congestion Control, Core-Stateless

ACM Reference Format:
Ferenc Fejes, Sándor Laki, Gergő Gombos and Szilveszter Nádas, Zoltán Kiss . 2019. Decoupling Delay and Resource Sharing Targets with Efficient Core-Stateless AQM. In *SIGCOMM '19: ACM SIGCOMM 2019 Conference, August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3342280.3342332

1 INTRODUCTION

Many novel applications cannot tolerate large delays and delay fluctuations, thus QoS solutions should contain mechanisms for ensuring controlled buffering delay. There is a rough consensus that any solution that keeps per-flow state inside the network will not scale with current demands. To overcome the scalability issues of flow-aware traditional QoS approaches, several core-stateless proposals [2, 7] have emerged in the past, but none of them can be viewed as a complete QoS solution, focusing on either providing fair share among flows or providing bounded delay. In core-stateless resource management, the operator policies are encoded into packet markings, resulting in a color or value marked on each packet. Then routers in the middle of the network operates solely based on packet markings with no flow-awareness. Most of the existing approaches couple resource sharing and delay targets, giving higher resource priority to traffic of smaller delay requirement. For example the packets of a real-time control flow have both high importance and short service requirements, while a gaming traffic having small delay requirement may be limited if its throughput demand is above a limit. In this case, if all the delivered packets will satisfy the delay requirement, but packets may be dropped.

The Per Packet Value (PPV) concept [3–6] provides a core-stateless way to control resource sharing among flows. Per subscriber and per-flow policies are applied by marking a continuous value called Packet Value (PV) on each packet. This marking can be done independently per subscriber. The schedulers in the network then aim at maximizing the total aggregate PV delivered without having any additional flow or policy information and using a single packet buffer.

Compared to [4] in which we demonstrate the resource sharing performance of PPV concept, the DPDK-Switch used in this demo can achieve much higher data rates, supports different delay targets and responsive TCP flows are used for the evaluation of the proposed delay-aware PPV AQM method. Accordingly, the method proposed in this paper also assumes that in addition to PV, each packet is marked with a delay class expressing its queuing delay requirement. The proposed AQM method decouples delay and resource targets by applying a PV-based drop policy and using delay classes to provide forwarded packets with guaranteed queueing delays. Note that a packet with small delay target can also be dropped if there are packets to be forwarded with higher PV (higher importance). We show that our implementation can realize delay and resource sharing targets at the same time running on an x86 machine with DPDK support.

2 DELAY-AWARE PPV AQM

Fig. 1 depicts the architecture of the delay-aware PPV scheduler that has been implemented in DPDK. It extends the implementation in [4, 5] for several delay classes. Based on the Delay Class dc, the FIFO queue Q_{dc} is chosen for a packet. The method then decides whether the incoming packet is enqueued or dropped, based on the queue lengths $|Q_i|$ and the Packet Value composition stored in the histograms $H_{IN,i}$, which maintains the total number of

SIGCOMM '19, August 19–23, 2019, Beijing, China

ACM ISBN 978-1-4503-6886-5/19/08…$15.00
https://doi.org/10.1145/3342280.3342332

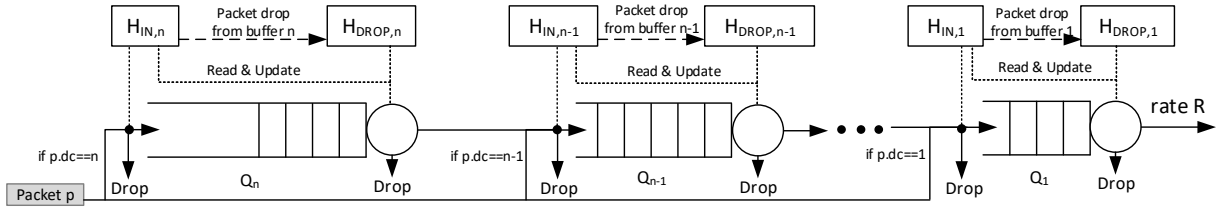

Figure 1: Delay-aware PPV AQM. Chained queues, one queue per Delay Class.

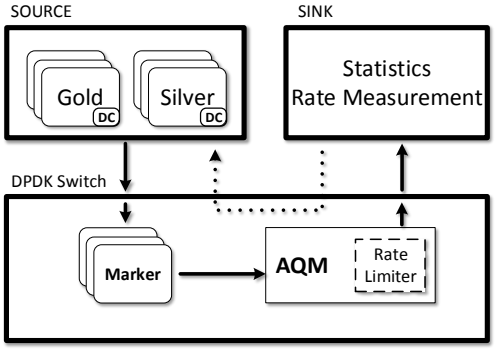

Figure 2: Demo setup

packet bits in the queues for all possible PVs. If the queue is full, packets already in the buffers Q_i, $i \leq dc$ with smaller PV than the one of the incoming packet needs to be dropped to make space, if possible. This is achieved by marking these packets to be dropped by moving bits from histogram $H_{IN,i}$ to a drop histogram $H_{DROP,i}$ that represents the packets to be dropped when they reach the head of the queue. If extra buffer space is available, dummy packets are inserted before the newly arrived packet, to avoid low-delay high priority packets unnecessarily triggering packet drop. Those dummy packets are always dropped when they reach the head of the first queue, Q_1. The maximum length of the queues is set such that $\sum_{i=1}^{j} |Q_i^{max}| = D_j \cdot R$, where D_j is the delay target of Delay Class j, and R is the system capacity. Whenever a new packet arrives to the system or a packet is served, the chained buffers are compressed towards the right. For this optimized implementation, packet dropping never happens mid-queue, only at queue borders, including before the server. The implementation by design meets the delay targets, because the amount of bits waiting before an already inserted packet never increases.

3 DEMO

We have implemented the proposed packet marking and delay-aware scheduling algorithms in DPDK. Our evaluation setup is depicted in Fig. 2, consisting of three nodes: 1) *Source node* that generates the traffic, 2) *DPDK Switch node* running our DPDK-based packet marker and scheduler, and 3) a *Sink node* where the traffic terminates. The source and sink nodes are connected to the switch via 40 Gbps links. For performance evaluation, we use both responsive TCP flows and unresponsive UDP traffic generated by the iperf traffic generator. Accordingly, all the traffic generated by the source node is sent to the DPDK Switch, where packets are first marked on distributed threads and scheduled on a single thread to the outgoing link limited to 10 Gbps, representing the bottleneck in the network. Note that each thread runs on a separate CPU core. Marker instances are created for each flow (modeling different subscribers) and the policy is assigned them in advance (Gold or Silver according to their source IP) while the delay class is encoded into the DSCP field of IP headers. After the packet value marking, all packets are sent to the same scheduler thread. After scheduling, the transmitted packets are forwarded to the sink node. Note that the router applies a simple forwarding on the reverse direction (no bottleneck in this direction). In the demo [1], we present various scenarios, where the number of Gold and Silver flows and their delay classes are varied. During the demo, flow and queue statistics are collected and stored with a time resolution of one second. The measurement time-series are shown in a real-time dashboard.

In the proposed scenarios, we define two policies: Gold and Silver and use three DCs (DC-1,DC-2 and DC-3) with delay requirements 5ms, 15ms and 30ms. The RTT is emulated by the Sink node and set to 10 ms. The number of flows are set to 200 (100 Gold - 100 Silver) generated by the iperf tool and the throughput statistics of 10-10 randomly selected flows are presented in the dashboard. To measure the end-to-end latency experienced by packets of different DCs, ping tool is used, periodically injecting ICMP Echo messages with Gold marking and different DCs into the system. The instantaneous queue size of DCs reported by the DPDK Switch is also displayed.

Scenario 1. We first present that latency and resource sharing are orthogonal requirements and can be decoupled with the proposed scheduler. All the Gold flows belong to DC-3 while the DC of Silver flows is chosen from DC-1 to DC-3. We show that the delay class has no effect on the resource share defined by the policy-based packet marking, solely affecting the experienced queueing delay.

Scenario 2. We extend the previous scenario with two selected users: 1) a malicious user that generates TCP traffic (large download), masquerading his traffic as gaming, thereby having DC-1 and Gold policy (small delay, best access). 2) A gamer user with small delay and low throughput requirements that generates low rate UDP traffic and is sensitive to packet losses and increased delay. The user's traffic is marked as Gold with DC-1. Note that both the malicious and gamer users insert their packets into the same (DC-1) fraction of the queue. This scenario demonstrates that even a malicious user who can give high priority to its traffic has no or only a slight effect on the performance of other flows including gaming that is sensitive to packet loss and delay.

Acknowledgement. The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013). The authors also thank the support of Ericsson.

REFERENCES

[1] 2019. Demo video. (2019). https://www.youtube.com/playlist?list=PL3GWppGvUS1p9oEu-2dORgBeo3pHZ-rRE

[2] Zhiruo Cao, Ellen Zegura, and Zheng Wang. 2005. Rainbow fair queueing: theory and applications. *Computer Networks* 47, 3 (2005), 367 – 392. https://doi.org/10.1016/j.comnet.2004.07.018

[3] Laki et al. 2017. Take Your Own Share of the PIE. In *Proceedings of the Applied Networking Research Workshop (ANRW '17)*. ACM, 27–32.

[4] S. Laki et al. 2018. Scalable Per Subscriber QoS with Core-Stateless Scheduling. In *ACM SIGCOMM 2018 Industrial Demos*.

[5] Sz. Nádas et al. 2018. Towards a Congestion Control-Independent Core-Stateless AQM. In *ANRW '18*. 84–90.

[6] Szilveszter Nádas, Zoltán Richárd Turányi, and Sándor Rácz. 2016. Per Packet Value: A Practical Concept for Network Resource Sharing. In *IEEE Globecom 2016*.

[7] Ion Stoica, Scott Shenker, and Hui Zhang. 2003. Core-stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High-speed Networks. *IEEE/ACM Trans. Netw.* 11, 1 (Feb. 2003), 33–46. https://doi.org/10.1109/TNET.2002.808414