A Core-Stateless L4S Scheduler for P4-enabled hardware switches with emulated HQoS

Ferenc Fejes**, Szilveszter Nádas*, Gergő Gombos**, Sándor Laki**
*Traffic Analysis and Network Performance Laboratory, Ericsson Research, Budapest, Hungary
**Communication Networks Laboratory, ELTE Eötvös Loránd University, Budapest, Hungary
{fejes, ggombos, lakis}@inf.elte.hu, szilveszter.nadas@ericsson.com

***Abstract*—Novel Internet applications often require low latency and high throughput at the same time, posing challenges to access aggregation networks (AAN). Low-Latency Low-Loss Scalable-Throughput (L4S) Internet service and related schedulers have been proposed to meet these requirements and also allow the coexistence of Classic and L4S flows in the same system. AANs generally apply Hierarchical QoS (HQoS) to enforce fairness among their subscribers. It allows subscribers to utilize their fair share as they desire, and it also protects traffic of various subscribers from each other. The traffic management engines of available P4-programmable hardware switches do not support complex HQoS and L4S scheduling. In this demo paper, we show how a recent core-stateless L4S Active Queue Management (AQM) proposal called VDQ-CSAQM can be implemented in P4, and executed in high-speed programmable hardware switches. We also show how a cloud-rendered gaming service benefits from the low latency and HQoS provided by our VDQ-CSAQM.**

I. Introduction

In the past years, novel applications such as AR/VR, cloud rendered gaming, HD or holographic video conferencing and remote presence have emerged, requiring high bandwidth, low latency or both. End users may have different subscriptions and Internet access. As gigabit-speed access links became widespread, the possibility of temporal and even permanent overloads in the AAN has increased. These periods can be handled by over-provisioning, but it has a high price: high infrastructure cost and underutilization in most of the time. HQoS is a widely adopted solution to ensure complex resource sharing policies in AANs, where resource sharing is controlled within and among traffic aggregates (TA), e.g., between operators, slices, users, and subflows of users. Though this solution is limited, nowadays, HQoS is typically enforced at the edge in network gateways, since all traffic going to the rest of the network has to flow through the gateway. L4S [1] introduces additional requirements on isolating L4S and Classic traffic and handling them differently which is not supported by existing HQoS solutions. In addition, it has been shown in [2] that a core-stateless resource sharing method called Hierarchical Per Packet Values (HPPV) can implement HQoS by only modifying its packet marking algorithms and the schedulers in the network remain the same. According to this concept, each packet is marked with a Packet Value (PV) expressing the importance of the given packet in the traffic mix at the edge of the network. The PVs are then used as an incentive in the scheduler of network nodes to decide which packet to forward and which packet to drop/mark with ECN Congestion Encountered (CE) flag. In this core-stateless approach, resource sharing policies between TAs and also within TAs can solely be defined by the packet marking strategy. A key advantage of this solution is that new policies can be introduced by only re-configuring the packet marking, keeping the schedulers unchanged. In [1], this core-stateless concept has been extended with the support of L4S requirements, resulting in a novel L4S scheduler called Virtual Dual Queue Core-Stateless Active Queue Management (VDQ-CSAQM) and a prototype implementation in DPDK. In this demo, we show that the lightweight design of VDQ-CSAQM enables its realization in hardware switches with P4-programmable data planes [3], only requiring a few simplifications and the careful separation of computational tasks between control and data planes. The resulting system can realize HQoS, meet the L4S requirements, and operate in high-speed P4-switches.

Application Domain Specific Highly Reliable IT Solutions project has been implemented with the support of the NRDI Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme. Supported by the ÚNKP-20-4 New National Excellence Program of the Ministry for Innovation and Technology from the source of the NRDI Fund.

II. P4 Implementation of VDQ-CSAQM (Fig. 1)

Data Plane (DP): To implement VDQ-CSAQM [1] on programmable hardware we simplified the per-packet operations as much as possible. As in the original design, two physical queues are configured with a strict priority scheduling between them. Non-ECT packets with packet value less then CTV_i (Congestion Threshold Value) are dropped in the ingress pipeline before queueing, while ECN CE marking happens at egress. We have replaced Virtual Queues (VQs) of the original algorithm with two groups of counters (counter overflow is allowed) for each queue: $arrivedBytes_i$ counts the amount of bytes arrived at Queue i, while $PvArrived_i$ counts the bytes carried in the packets with a given PV that arrived at Queue i. The PV range is limited to 1024 different values (encoded into 10 bits), which corresponds to a precision of 3% in the range of 10 kbps to 1 Tbps.

Control Plane (CP): The coupling of the VQs is implemented in the CP by also considering the counters of Queue 0 for Queue 1. CP syncs $PvArrived_i$ from the DP in every 50 ms that is used to calculate the Packet Value histograms for both queues. The DP sends the $arrivedBytes_i$ counters as a digest

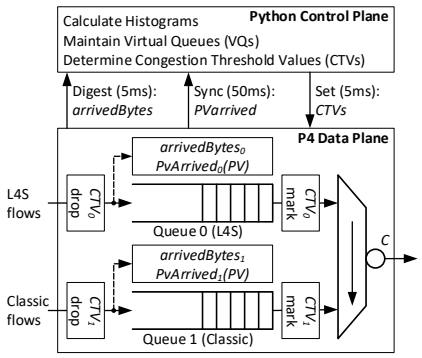

Figure 1. P4 VDQ-CSAQM Implementation

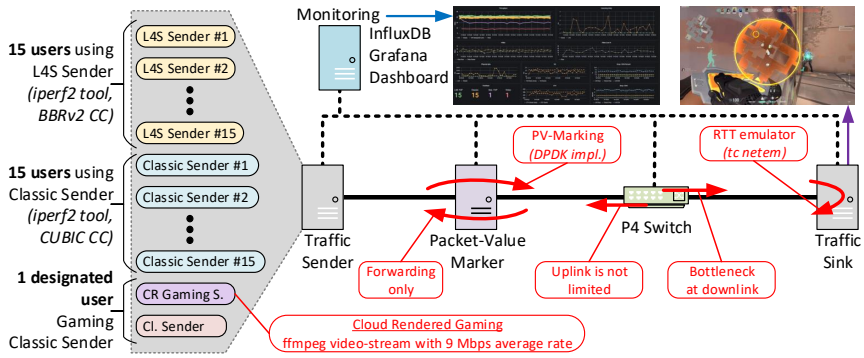

Figure 2. Overview of the demo setup

message to the CP every 5 ms. Using these counters and their history the CP maintains the two VQs. Based on the length of the VQs we calculate a probability $q_i = VQ_i^{target}/VQ_i$ for marking packets ($p = 0$, if the VQ length VQ_i is below the target). The new packet value threshold CTV_i is calculated as the percentile of histogram i at q_i.

III. Demo Scenarios

Fig. 2 illustrates our demo setup with a testbed consisting of five nodes, each link between the nodes has 40 Gbps capacity. **1)** The *Traffic Sender* generates the test traffic: 15 L4S and 15 classic background users are represented by 1 TCP flow per user. An additional designated user sends a classic TCP and/or a Cloud Rendered Gaming video flow. The gaming flow is sent over UDP and it is not congestion controlled. It has 9 Mbps average rate and temporal bursts can reach 29 Mbps peak rate on 200 ms timescale. **2)** The *Packet-Value Marker* runs our DPDK implementation of simplified HPPV. By default it assigns the same policy for all users (both background and designated) representing *equal sharing for all users*. The traffic class (L4S or Classic) is encoded to the DSCP field. The gaming traffic is classified as L4S therefore it experiences very small buffering delay; and it is non ECN capable, thus it might be dropped in the AQM. In the *HQoS* case, the subflows of the designated user are also identified and controlled by the HPPV marker. **3)** The *P4 Switch* (Stordis BF2556X-1T based on Tofino ASIC) implements VDQ-CSAQM as highlighted in Section II, using the default parameters of [1]. Its outgoing port towards Sink node is rate limited at 1 Gbps, emulating the bottleneck in the AAN. **4)** The *Traffic Sink* terminates the traffic of the users. It also emulates the propagation RTT (5 ms) by delaying TCP acknowledgments. **5)** The *Monitoring* node collects the operational logs from the other nodes, stores them in a time-series database (InfluxDB) and visualizes them on a dashboard (Grafana) in real-time.

The **demo as shown in the video at [4]** covers three scenarios: **1) Equal sharing for all users and a designated user with gaming traffic only**: The packet value marking ensures that users share the bottleneck capacity equally, resulting in about 32 Mbps capacity share for each user. As a result, the gaming traffic does not experience loss. This scenario also shows that using CSAQM the unused resources can be occupied by other flows, but if the rate of the gaming video is increasing, CSAQM still saves the video stream from packet drops. The experienced queueing delay is almost zero for L4S and small for Classic flows. **2) Equal sharing for all users and a designated user with both gaming and classic traffic**: The scenario demonstrates that resource sharing among end-users is not enough to provide good quality of experience since other traffic of the same user may also affect the performance of loss-sensitive flows. The presence of a parallel classic flow results in approx. 10-15% loss ratio in the gaming traffic, significantly reducing the quality of experience in case of delay sensitive, real-time applications like cloud rendered gaming. The delay of the gaming traffic remains low. **3) HQoS and a designated user with both gaming and classic traffic**: We use the previous scenario with a different packet value-marking strategy, emulating hierarchical quality of service (HQoS). In addition to the resource sharing policy among end-users, a second policy-level is introduced, expressing a 4:1 weighting between the gaming and TCP traffic, respectively. The weighting is solely implemented in the Marker node by assigning larger PVs to packets of the gaming traffic with higher probability in a way that the overall PV distribution of the designated user remains unchanged. This means that the rate of gaming traffic can be increased up to $4/5^{th}$ of the user's capacity share without experiencing any packet loss. We configured this high weight for gaming, because it is necessary to avoid packet loss of its video stream even during its peak periods. The lossless transmission is realized and it results in good QoE for gaming. At the same time, the TCP session of the designated user can utilize the rest of the user's fair share despite the burstiness of the gaming video traffic. For video streams with more stable sending rates even smaller share of the second policy-level weights would be adequate.

Note that resources are not reserved in advance and policies are not communicated to the P4 Switch at all. Reconfiguration of QoS policies only requires changes in the packet marking.

References

[1] S. Nádas *et al.*, "A congestion control independent L4S scheduler," in *ACM/IRTF ANRW*, 2020, pp. 45–51.

[2] S. Nádas *et al.*, "Stateless resource sharing in networks with multi-layer virtualization," in *ICC*. IEEE, 2019, pp. 1–7.

[3] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM CCR*, vol. 44, no. 3, pp. 87–95, 2014.

[4] F. Fejes *et al.*, "Demo video," in *http://ppv.elte.hu/ic21*, 2021-01.