Resource Sharing Beyond FQ: 35K Users at 100Gbps

Dávid Kis, Gergő Gombos, Sándor Laki
ELTE Eötvös Loránd University
Budapest, Hungary
lakis@inf.elte.hu

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

ABSTRACT

Core-stateless resource sharing solutions implemented in P4 hardware data planes have been proposed in the past few years. They share the idea of tagging packets with special values at the network edge that are then solely used for deciding how to handle packets in the network in case of congestion. Though the scheduler of our Core-Stateless Active Queue management (CSAQM) was implemented in P4 and was evaluated on Intel Tofino ASIC, the packet marker have only had a DPDK-based software implementation so far. In this demo, we present the full data plane implementation of CSAQM. Both packet marking and packet scheduling are executed by an Intel Tofino ASIC. We demonstrate the scalability of our implementation by showing policy enforcement among up to 35000 subscribers at a 100 Gbps bottleneck using only a single queue. In addition, we also present the resource sharing and isolation properties of CSAQM between flows with different rate control strategies, resulting in flow-specific congestion signals (drop probabilities) by design.

CCS CONCEPTS

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

KEYWORDS

Resource Sharing, QoS, Congestion Control, Core-Stateless

ACM Reference Format:
Dávid Kis, Gergő Gombos, Sándor Laki and Szilveszter Nádas . 2022. Resource Sharing Beyond FQ: 35K Users at 100Gbps. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22 Demos and Posters), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3546037.3546045

1 INTRODUCTION

The increasing broadband access speeds of mobile and fixed networks lead to new challenges in the transport Access-Aggregation Network (AAN). The AAN may itself become a bottleneck of transmission instead of access links. When a high-speed link is shared among thousands of subscribers without any isolation mechanisms, the resulting resource sharing depends on the number of flows of each subscriber and the congestion controls used. Fair Queueing (FQ) scheduler may be used to enforce fairness among the users [7] in theory, but it does not scale as the number of subscribers grow as it requires many queues to be maintained.

Core-Stateless Active Queue management (CSAQM) [5] and other recent proposals [8] exploit the benefits of programmable data planes to expand traffic management capabilities. In CSAQM, each packet is tagged at the edge of the network with a Packet Value (PV). The PVs are then used as an incentive in the scheduler of network nodes to decide which packet to forward, drop or ECN mark. Resource sharing policies are solely defined by the packet marking strategy, which is described by bandwidth-function-like policies [4]. A key advantage of this solution is that new policies can be introduced by only re-configuring the packet marking at the network edge, keeping the schedulers unchanged. We have already demonstrated that a CSAQM scheduler can be implemented on P4-programmable devices [1, 2], but the packet marking component has only been implemented in a DPDK-based software till now. Its P4 implementation would further simplify the deployment.

2 DATA PLANE DESIGN FOR SCALABLE PACKET MARKING

In CSAQM, the resource sharing policy to be applied for each traffic aggregate is implemented by a dedicated packet marker instance. Each such component continuously measures the arrival rate (R_j) of jth traffic aggregate, chooses a sample between 0 and R_j uniformly at random and applies the policy function $v(.)$ at this random value to obtain a PV for tagging the packet [5]. Even though this is a very simple mechanism, implementing it on a P4 programmable ASIC is still challenging. While Tofino hardware allows generating random integers between 0 and $2^n - 1$, converting them into a range between 0 and an arbitrary integer value required some special trick. Our simplified packet marking algorithm for Tofino ASIC is depicted in Fig. 1. The packet processing pipeline starts with a table responsible for maintaining subscriber rates and setting the policy to be applied. This table matches on a subscriber identifier (e.g., IPv4/IPv6 address) and each packet updates a rate measurement instance R implemented as a low-pass filter stateful object. We assume that the policy function has an exponential decay and thus an exponential binning of the throughput-axis can be used to discretize this function. As shown in the figure, bin sizes are equidistant on a logarithmic scale and each bin (e.g., Δ_i from a^{i-1} to a^i) represents a unique packet value-level marked with red lines. To support this compact representation, we first convert rate R to a bin index i using a logarithm table. Table RateIndex applies range matches on the subscriber's rate R and results in i so that $R \in \Delta_i$. The uniform random sample between 0 and R is approximated with a logarithmization trick. We first take a random value rnd from range 0 to 255 and then compute the bin index of $\frac{rnd}{255} \times R$, using table RandomRateEstim. Instead of directly computing the rate sample, we take the logarithm of it and offset the bin index

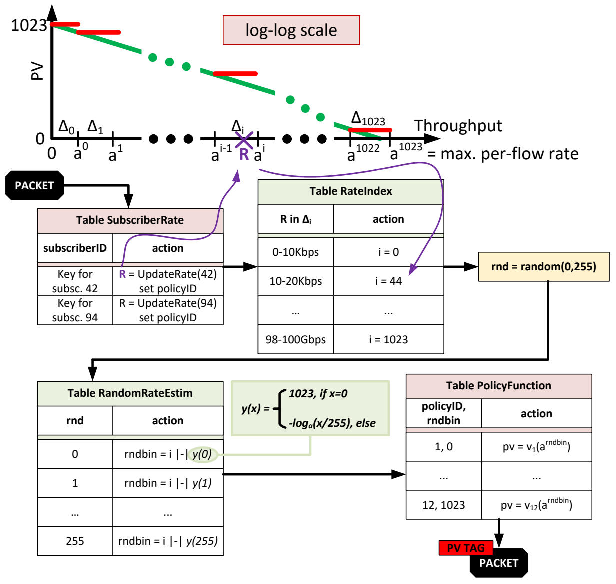

Figure 1: Data Plane Design of Packet Marking

i accordingly, resulting in a random index *rndbin*. The offset is shown as $y(x)$ which is precomputed and hard-coded in the table. Finally, we determine the packet value to tag the packet by applying table PolicyFunction that stores the discretized representations of policy functions. The proposed design requires limited number of per-subscriber stateful data plane resources and, with the current prototype implementation, can scale up to 35K subscribers. This limitation is caused by our preliminary rate measurement algorithm that relies on an array of low-pass filter externs having as many elements as the number of subscribers to be distinguished. This step of our P4 pipeline is mapped to a single stage, reaching the per-stage resource limit. However, scaling beyond 35K subscribers is possible by optimizing the resource utilization of our pipeline algorithm (e.g., mapping low-pass filter externs to more stages). We considered typical policy functions and PV encoding schema during the data plane design of our packet marker. For this case, apart from the different rate measurement algorithms, the P4 pipeline is equivalent to our previous DPDK implementation.

3 DEMO SCENARIOS

Our evaluation setup is depicted in Fig. 2. We perform CSAQM packet marking in the data plane of a Tofino-based P4-switch. Packet marking can be omitted for non-edge bottlenecks. The 100 Gbps bottleneck has been created by interconnecting two ports of the switch. The bottleneck is managed by the CSAQM scheduler that has both data and control plane components running on the same P4-switch. Thousands of UDP flows emulating subscribers' traffic are generated with the DPDK PktGen tool on a source connected with two 100 Gbps links to the switch. The throughput of these flows is measured using our DPDK-based collector program on a sink machine connected with a single 100 Gbps link. The other source and sink machines run iperf2 [6] to generate the traffic of designated subscribers and are connected with a 40 Gbps link each.

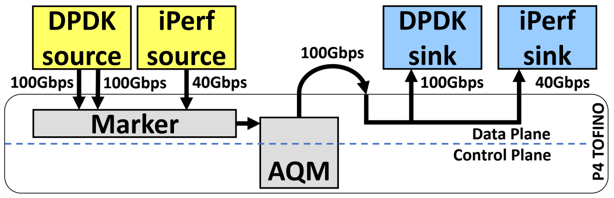

Figure 2: Demo setup

Measurements are collected from both sink nodes and the P4-switch and visualized on a Grafana-dashboard in real-time. Each user is marked by a Silver or a Gold policy, which are configured to achieve 1:3.4 sharing between Silver and Gold users, resp.

Our demo video [3] covers the following four scenarios. **1) 10k Gold and Silver subscribers:** The DPDK source generates the traffic of 8000 Silver users on one of the interfaces and 2000 Gold users on the other, resulting in 200 Gbps load on the bottleneck. The applied policies result in 21.8 and 6.4 Mbps average per-user throughput for the two classes. The virtual queue of CSAQM discards 51.5% of the incoming packets according to the resource sharing policy represented by the packet marking. This scenario shows the resource sharing property of our packet marker and scheduler in case of high, but stable traffic loads. **2) Designated users with different demands:** We add two Gold UDP flows (as 2 designated users) generated by iperf to the system, one sending with 15 Mbps, the other with 50 Mbps. No packets of the 15 Mbps flow are lost, as the user's demand is below its fair share (of 21.8 Mbps). The user with 50 Mbps traffic load experiences 58% loss, which is higher than the average loss rate in the AQM, and thereby it reaches the same throughput as other Gold subscribers. This scenario demonstrates that users demanding less than their fair share are not affected by other subscribers' high loss rate. **3) Designated users with congestion controlled flows:** We add 10 Gold Cubic TCP flows to the mix, emulating 10 additional users with congestion controlled flows. Their throughput is $\approx$ 18.5 Mbps, somewhat below the UDP-based other Gold users. The packet loss of these users is about 2%. We show that the congestion control of a single TCP flow cannot perfectly utilize the Gold fair share. We believe that the rate measurement algorithm can be further tuned to optimize the TCP performance, which we leave for future work. Rate measurement for the constant bitrate UDP traffic is naturally much simpler. Our method results in traffic/congestion control-specific loss rates, enabling the coexistence of even incompatible rate control mechanisms in the same system. **4) Scaling up to 35k subscribers:** In this case, we only use our DPDK source to generate the traffic of 7000 Gold and 28000 Silver users. The results show that the algorithm still maintains the desired resource sharing even at this scale.

Acknowledgement. S. Laki and G. Gombos thank the support of the "Application Domain Specific Highly Reliable IT Solutions" project that has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme. The authors also thank the support of Ericsson.

REFERENCES

[1] Ferenc Fejes, Szilveszter Nádas, Gergő Gombos, and Sándor Laki. 2021. A Core-Stateless L4S Scheduler for P4-enabled hardware switches with emulated HQoS. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM Demo)*. IEEE, 1–2.

[2] Ferenc Fejes, Szilveszter Nádas, Gergő Gombos, and Sándor Laki. 2022. DeepQoS: Core-Stateless Hierarchical QoS in Programmable Switches. *IEEE Transactions on Network and Service Management* (2022).

[3] Dávid Kis, Gergő Gombos, Sándor Laki, and Szilveszter Nádas. 2022-05. Demo video. https://www.youtube.com/watch?v=c0OWlsp6flg

[4] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. 2015. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In *Proceedings of ACM Sigcomm* (London, United Kingdom) *(SIGCOMM '15)*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/2785956.2787478

[5] Sándor Laki, Szilveszter Nádas, Gergő Gombos, Ferenc Fejes, Péter Hudoba, Zoltán Turányi, Zoltán Kiss, and Csaba Keszei. 2020. Core-Stateless Forwarding With QoS Revisited: Decoupling Delay and Bandwidth Requirements. *IEEE/ACM Transactions on Networking* 29, 2 (2020), 503–516.

[6] Robert McMahon, Battu Kaushik, and Tim Auckland. 2005. Iperf: The TCP/UDP bandwidth measurement tool. https://sourceforge.net/projects/iperf2/

[7] M. Shreedhar and George Varghese. 1995. Efficient Fair Queueing Using Deficit Round Robin. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (Cambridge, Massachusetts, USA) *(SIGCOMM '95)*. Association for Computing Machinery, New York, NY, USA, 231–242. https://doi.org/10.1145/217382.217453

[8] Zhuolong Yu, Jingfeng Wu, Vladimir Braverman, Ion Stoica, and Xin Jin. 2021. Twenty Years After: Hierarchical Core-Stateless Fair Queueing. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 29–45.