Scalable Per Subscriber QoS with Core-Stateless Scheduling

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

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

Csaba Keszei
Ericsson
Santa Clara, CA, USA
Csaba.Keszei@ericsson.com

ABSTRACT

In broadband access networks operators aim to provide fairness among different subscribers. It cannot be provided by commodity network switches, thus Broadband Network Gateways (BNG) implement Hierarchical Quality of Service, emulating network bottlenecks and performing per subscriber weighted fair queueing at the entry point of the access network. As the number of subscribers increases to several thousands this solution scales badly. It needs at least one queue per subscriber, resulting in high memory footprint, memory management problems and processing intensive scheduling. Though it can still be implemented using costly, specialized hardware, it is harder and harder to implement it in cloud-native routers running on commodity hardware. In this demo paper, we propose a highly-scalable solution for this problem that can even work in cloud-native environments. The proposed core-stateless method is based on the the Per Packet Value concept, where edge nodes assign Packet Values calculated based on operator policy per subscriber. These markings are then used for packet scheduling without the need of additional policy or flow information. This demo shows how the PPV concept can provide efficient and more scalable alternative to existing HQoS solutions while implementing the same rich policies. The processing and memory footprint of the proposed scheduler is independent of the number of subscribers, while the per subscriber packet markers can be distributed among virtual machines as they operate independently.

1 INTRODUCTION

Standardized, deployed commodity network switches cannot usually provide per subscriber fairness. In many cases this is acceptable, however in access networks, specifically fixed broadband ones, per subscriber fairness usually needs to be enforced. In addition, a subscriber may have several internal priority levels to be taken to account. To achieve this Broadband Network Gateways (BNG) implement per subscriber Hierarchical Quality of Service (HQoS): several queues are assigned to each subscriber and a scheduler (e.g. Weighted Fair Queueing) is applied to them [4]. As the number of subscribers, sub-classes and link capacities increase, HQoS requires more and more processing and memory resources. Though it can still be implemented using costly, specialized hardware, it is harder and harder to implement it on cloud routers using commodity hardware.

The Per Packet Value (PPV) concept [3, 5] 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 the total aggregate PV delivered without having any additional flow or policy information and using a single packet buffer.

In this demo we show how our PPV implementation on a virtual router product scales well with the increasing number of users, while it provides the desired resource sharing among subscribers.

2 PPV-BASED BNG IMPLEMENTATION

Fig. 1 depicts the architecture of the optimized Packet Value-based scheduler [5] that has been implemented in a cloud-native virtual BNG product. At *Packet Arrival* the method first decides whether the incoming packet is enqueued or dropped, based on the *maximum queue length* and the Packet Value composition stored in the histogram H_{IN} which maintains the total number of packet bits in the queue for all possible PVs. If the queue is full, packets already in the buffer with smaller PV than the one of incoming packet needs to be

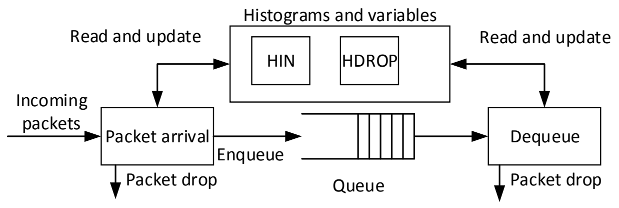
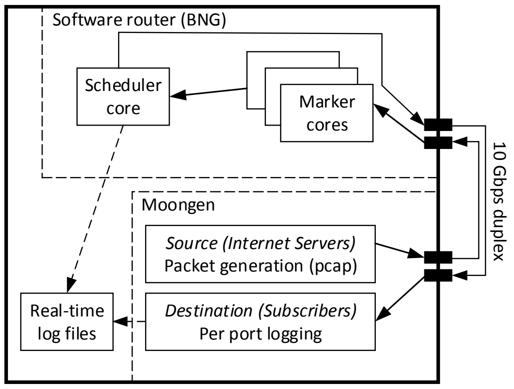
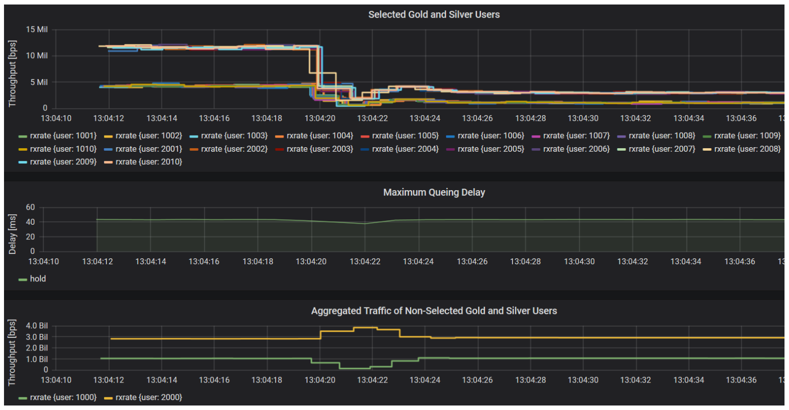


Figure 1: Scheduler Implementation



(a)



(b)

Figure 2: Demo architecture (a) and measurements (b)

dropped to make space, if possible. This is achieved by marking these packets as dropped by moving bits from histogram H_{IN} to a drop histogram H_{DROP} that represents the packets to be dropped at dequeue phase. This behavior avoids looking at or dropping packets mid-queue. At *Dequeue* phase, the scheduler drops packets based on H_{DROP}, if $H_{DROP}(PV) > 0$, while it is possible, then serves the first packet. The optimized implementation only drops packets upon arrival or at the front. Further we maintain the minimum PV in H_{IN} to be able to decide fast whether to drop an incoming packet.

3 DEMO

We have implemented packet marking and scheduling algorithms in a cloud-native virtual router implementing a BNG [4] network function. Our prototype implementation is a drop-in replacement of the existing Deficit Round Robin-based Traffic Management (TM) component of the router. The virtual router runs on an Intel(R) Xeon(R) CPU E5-2630 @ 2.30GHz host PC with 32 GByte RAM. Moongen [2] is used to generate the traffic for a changing number of subscribers, always sending with 10 Gbps in total. Both the router and the traffic generator are running on the same PC and they are connected using a physical 10 Gbps loop between the two ports of an Intel Niantic (82599) card as shown on Fig. 2a. The traffic is sent to the software router, where packets are first marked on distributed cores and scheduled on a single core to 4 Gbps, representing the bottleneck in the access network. Marker instances are created for each destination IP address (modeling different subscribers). After the packet value marking, all packets are sent to the same scheduler core. After scheduling, the transmitted packets are sent back to Moongen, where throughput statistics for 10-10 selected Gold and Silver subscribers and the aggregated values are collected. The queue length of the scheduler core of the router is also queried in every second. The measurement time-series are then shown in a real time dashboard illustrated in Fig. 2b.

In the proposed scenarios we define two policies: Gold and Silver. The number of subscribers starts at 512 (256 Gold - 256 Silver), we then increase those to 2000 (1000-1000) and to 4000 (2000-2000). Two different traffic management (TM) policy sets are defined: TM-1) until Silver subscribers receive 1 Mbps, all subscribers shall get the same throughput. Above that Gold subscribers shall receive twice the throughput of Silver ones. TM-2) the minimal Silver throughput is set to 128 kbps, which means that in all investigated cases Gold subscribers shall receive twice the throughput. We measure the throughput of 10-10 selected Gold and Silver subscribers, 5-5 of those send with twice the speed of other subscribers to demonstrate that even when they send with higher speed they receive the same throughput. We also measure the bulk of the rest of Gold and Silver subscribers aggregates. These measurements are shown on our example video [1]. Fig. 2b depicts the throughput of the selected subscribers, the bulk throughput and the queue length, when the number of subscribers is changing from 500 to 2000. As we increase the number of subscribers it is visible how each subscriber, and the aggregates, get a throughput close to its fair share. Meanwhile, the length of the scheduler queues remains in the same region.

REFERENCES

[1] 2018. Demo video. at sigcomm18indus-demos.hotcrp.com. (2018).
[2] Emmerich et al. 2015. Moongen: A scriptable high-speed packet generator. In *Proc. of the 2015 IMC.* ACM, 275–287.
[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] Csaba Keszei. 2017. Make DPDK's software traffic manager a deployable solution for vBNG. In *DPDK Summit.*
[5] Szilveszter Nádas, Zoltán Richárd Turányi, and Sándor Rácz. 2016. Per Packet Value: A Practical Concept for Network Resource Sharing. In *IEEE Globecom 2016.*