T4P4S & PIE: Towards an AQM Evaluation Testbed with P4 and DPDK

Authors: Péter Vörös, Sándor Laki Affiliation: ELTE Eötvös Loránd University, Budapest, Hungary
On site presenters: Péter Vörös (vopraai@inf.elte.hu), Sándor Laki (lakis@elte.hu)

We are living in an Active Queue Management (AQM) renaissance mainly due to overly large buffers and thus large queueing delays, known as the bufferbloat problem. In addition to classical AQM schemes like RED, delay-aware approaches like CoDel, PI controller-based approaches like PIE and PI2 have also been proposed in the past decade. Active Queue Management (AQM) addresses the problem arising from using unnecessarily large and managed buffers and thus it aims at improving network and application performance. These methods introduce different drop policies to proactively drop packets according to queue states and parameters. In the P4-Arch TM SubWG it has been shown that drop policies of AQM methods can be implemented in ingress and/or egress control blocks of a P4 program. The correctness of the P4 AQM implementations has already been shown in simulations using NS-3 with BMv2 [1]. In this demo, we go one step further: demonstrating PIE AQM [5] with our DPDK-based compiler called T4P4S [3,4] (tapas) in a testbed with realistic traffic mixes.

PIE AQM applies a PI controller to keep the queuing delay close to a target value by modifying the applied drop probability. Though the calculation of drop probability is complex, it only has to be done periodically (once in a delta interval) and not for each packet. PIE is implemented as an ingress process, applying packet drop before enqueueing the packet. In this demo, we will use our simplified P4 implementation of PIE [2].

To support AQM evaluation T4P4S compiler has also been modified. The run-to-completion model of T4P4S has been replaced by splitting the pipeline into two parts: 1) from parsing to ingress control and 2) egress control to deparsing. The two parts can be assigned to different CPU cores interconnected with queue (ring buffer). Note that currently T4P4S uses the v1model as its architecture. In addition to v1model we also introduced a new metadata field for providing queue latency information in the ingress control block. T4P4S will also provide real-time information on the queue states and the total throughput achieved. The architecture can be seen in Fig. 1, consisting of a T4P4S switch and a traffic generator node. The test traffic will be generated by the traffic generator node using iperf for responsive traffic mixes. The number of sources will also be varied during the experiment. A bottleneck (BN) will be introduced at the outgoing interface of T4P4S switch in the downlink direction. We will also show real-time statistics including total throughput, queueing delay and number of packet drops like in Fig. 2.

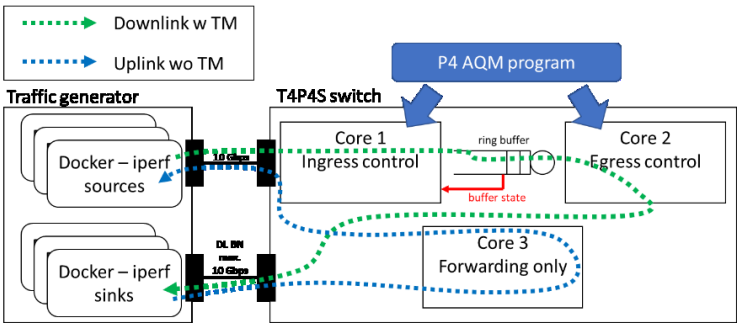

Fig. 1: Demo setup; green – DL traffic, blue – UL traffic

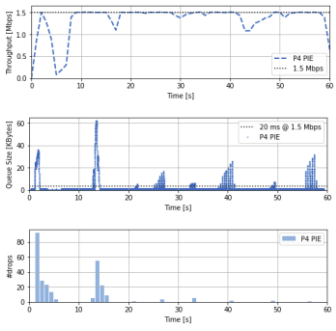

Fig. 2: Collected statistics

References

[1] NS-3-BMv2: url: https://github.com/PIFO-TM/ns3-bmv2
[2] PIE implementation in P4: https://github.com/PIFO-TM/ns3-bmv2/blob/master/traffic-control/examples/p4-src/pie
[3] T4P4S-16, url: https://github.com/P4ELTE/t4p4s/tree/t4p4s-16
[4] P. Vörös, D. Horpácsi, R. Kitlei, D. Leskó, M. Tejfel, S. Laki: „T4P4S: A Target-independent Compiler for Protocol-independent Packet Processors”, IEEE HPSR 2018, June 17-20, Bucharest, Romania
[5] Pan, Rong, et al. "PIE: A lightweight control scheme to address the bufferbloat problem." 2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR). IEEE, 2013.