Towards an AQM Evaluation Testbed with P4 and DPDK

Sándor Laki, Péter Vörös, Ferenc Fejes
ELTE Eötvös Loránd University, Budapest, Hungary
E-mail: {lakis,vopraai,fejes}@inf.elte.hu

Standard loss-based TCP's congestion control plus large unmanaged buffers in Internet routers, switches, device drivers, etc. cause the problem called **bufferbloat,** leading to **latency issues** for interactive/multimedia applications. To solve the problem **Active Queue Management** (AQM) tries to signal the onset of congestion by **dropping or ECN marking packets.** AQMs have three main goals: *1) Maintaining low average queue/latency, 2) Allowing occasional packet bursts, 3) Breaking synchronization among TCP flows.*

In this demo:

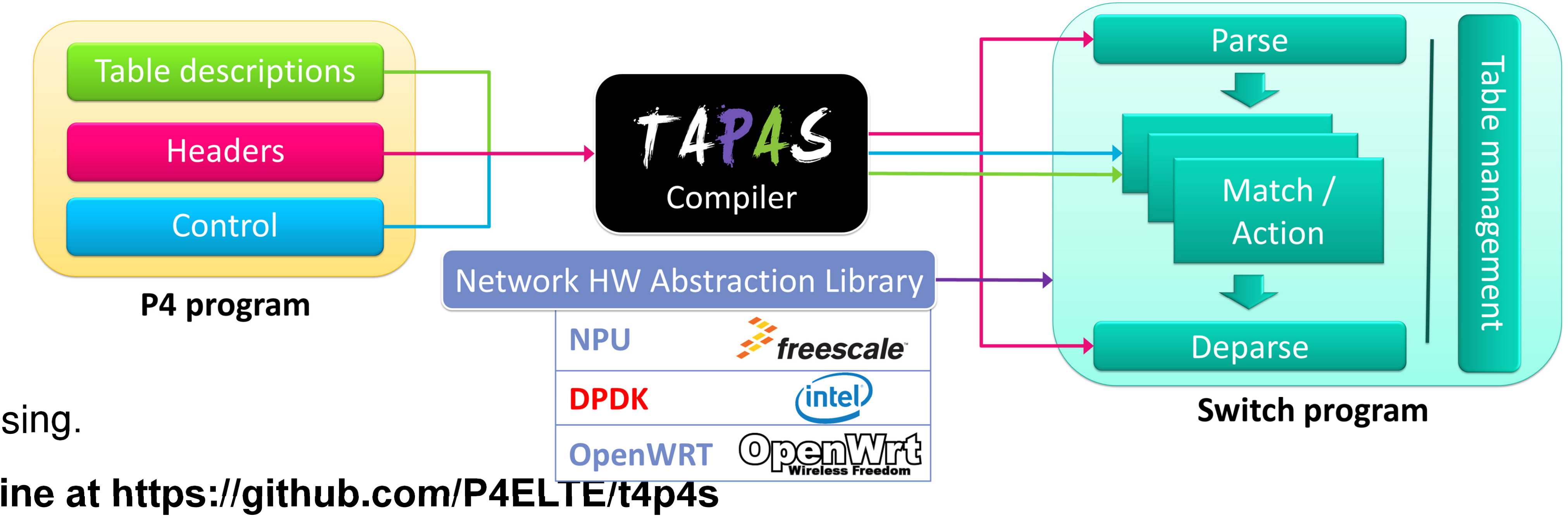
- We **demonstrate that AQM** algorithms can be described in P4 using an architecture model extended by access to queue states, implementing a simple **FIFO as reference, RED and PIE AQMs in P4**.
- We propose **a unified evaluation framework** that uses P4 as a common language to **describe AQM algorithms** and enables the evaluation with **good coverage of possible parameters** including unresponsive traffic, responsive different with various congestion controls, number of flows (up to 10Ks), different bottleneck capacities, etc.
- We provide **a high-speed prototype** based on our P4 compiler for DPDK (T4P4S).
- **Queue and flow statistics** are **monitored**, **stored** in an InfluxDB and **visualized** in a Grafana dashboard **in real-time**.

T4P4S – AN OPEN SOURCE P4 COMPILER FOR DPDK

T4P4S - Translator for P4 Switches turns a P4 code into a target independent C core program running on the top of a **Network Hardware Abstraction Library** (NetHAL).

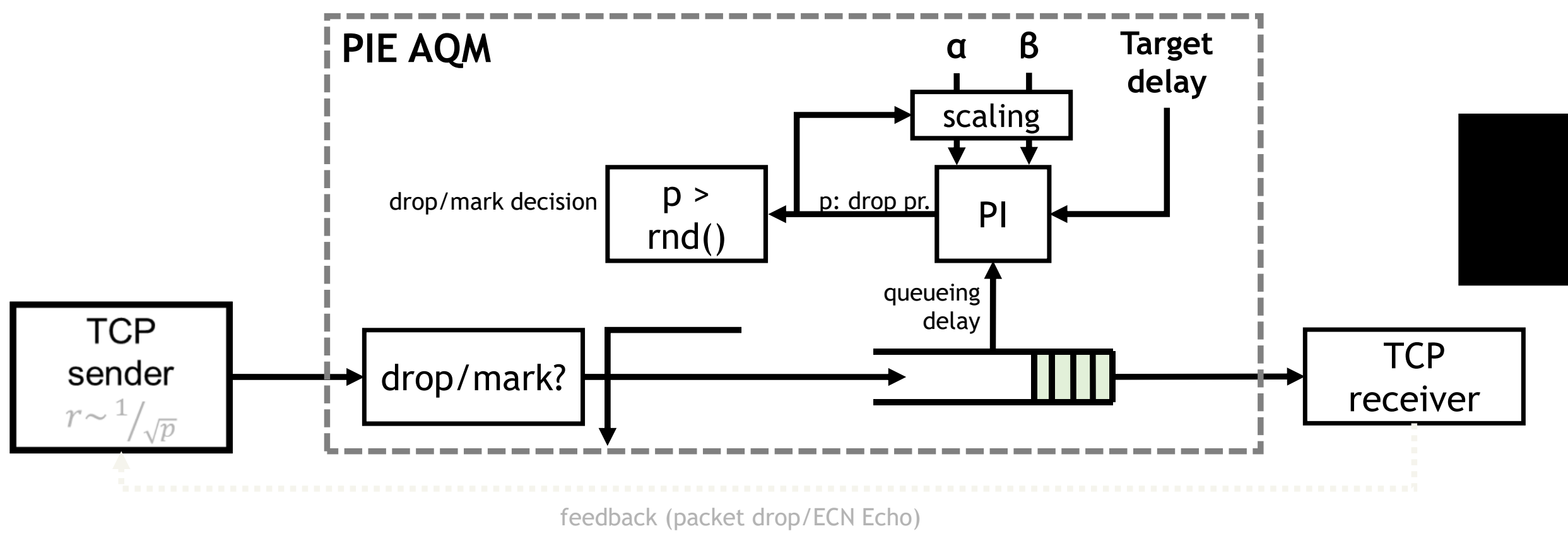
NetHAL is currently available for DPDK, ODP and Native Linux targets. To run the core program on a specific hardware the appropriate NetHAL needs to be linked.

The compiled switch program then parse incoming packets, apply match-action rules and deparse messages before egressing.

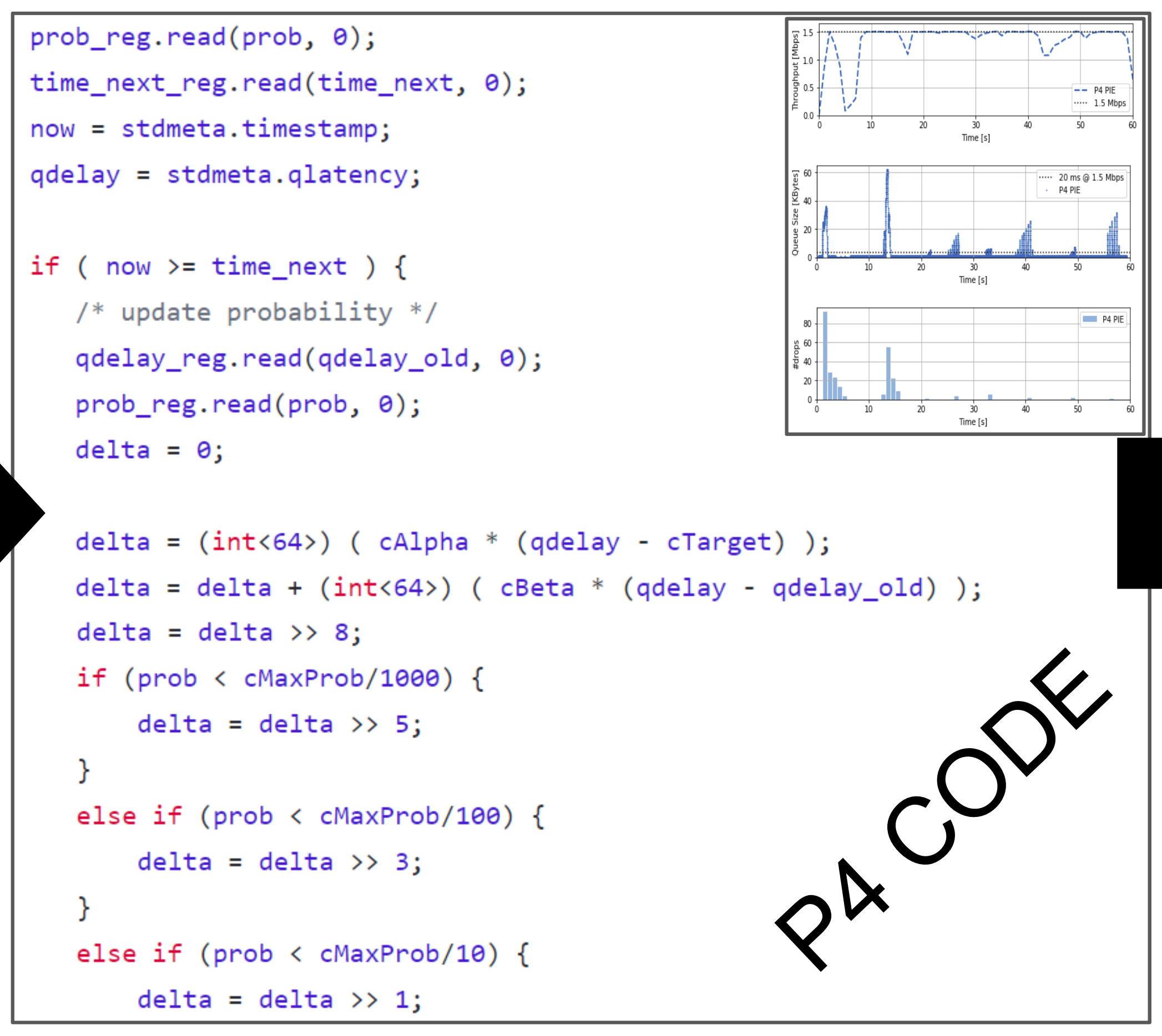


Available online at https://github.com/P4ELTE/t4p4s

DROP POLICIES IN P4
AN EXAMPLE: PIE AQM

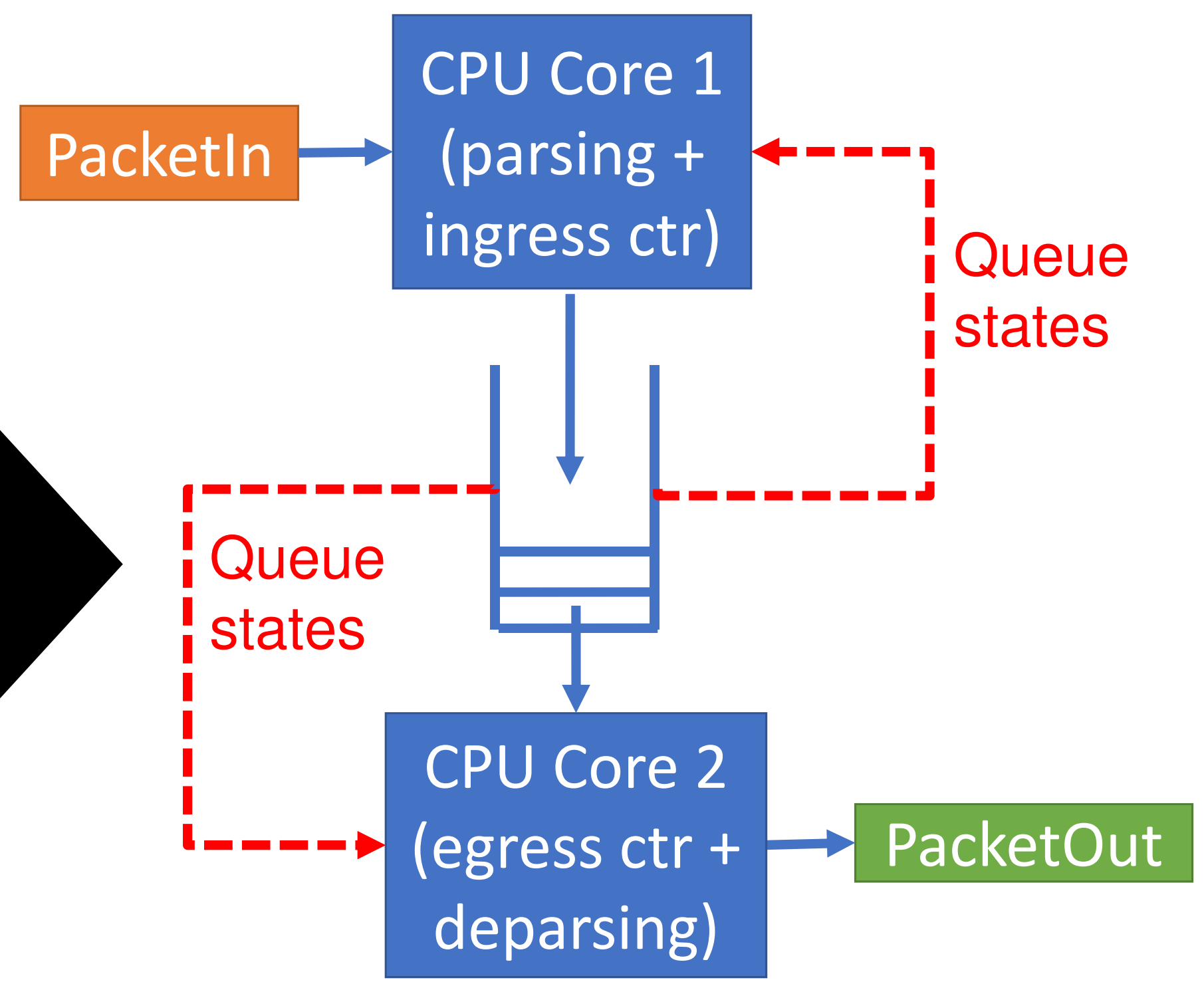


AQM ALGORITHM



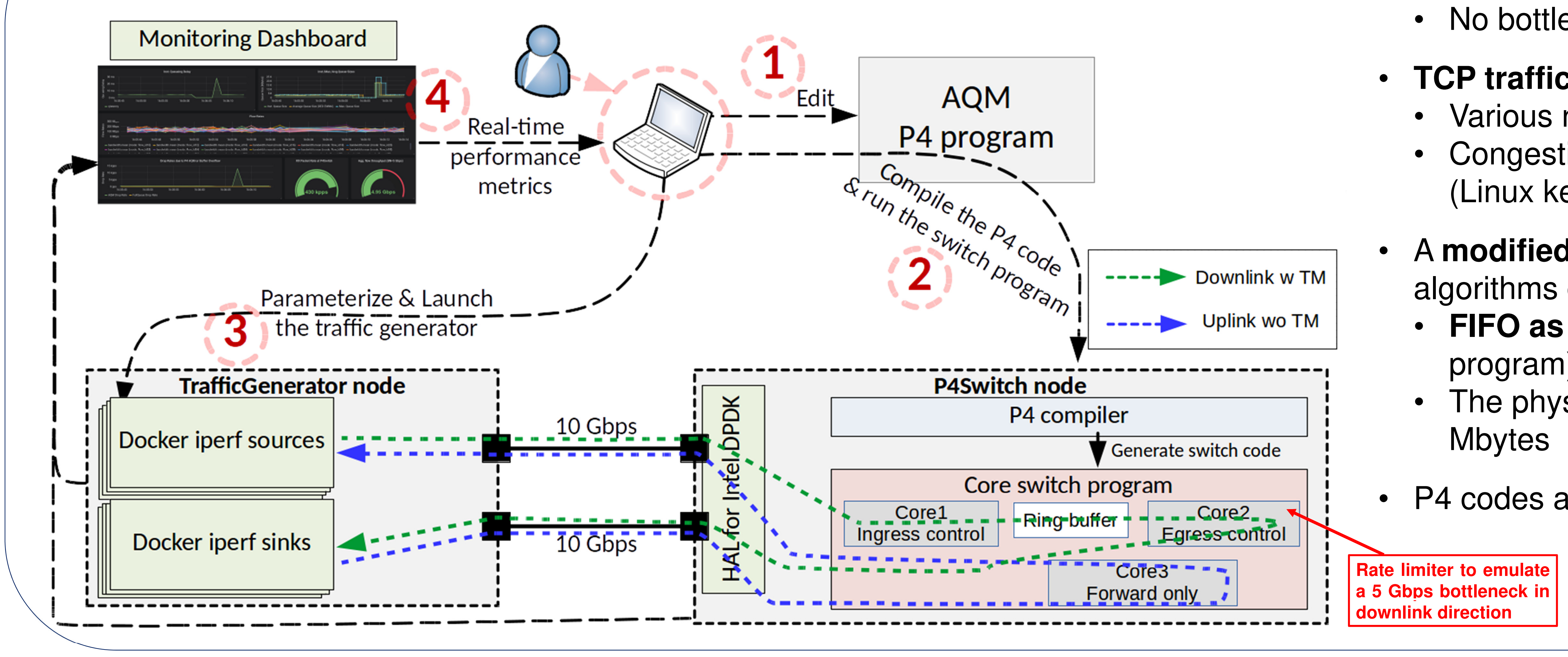
```
prob_reg.read(prob, 0);
time_next_reg.read(time_next, 0);
now = stdmeta.timestamp;
qdelay = stdmeta.qlatency;

if ( now >= time_next ) {
   /* update probability */
   qdelay_reg.read(qdelay_old, 0);
   prob_reg.read(prob, 0);
   delta = 0;

   delta = (int<64>) ( cAlpha * (qdelay - cTarget) );
   delta = delta + (int<64>) ( cBeta * (qdelay - qdelay_old) );
   delta = delta >> 8;
   if (prob < cMaxProb/1000) {
       delta = delta >> 5;
   }
   else if (prob < cMaxProb/100) {
       delta = delta >> 3;
   }
   else if (prob < cMaxProb/10) {
       delta = delta >> 1;
```

P4 CODE


PacketIn
CPU Core 1 (parsing + ingress ctr)
Queue states
Queue states
CPU Core 2 (egress ctr + deparsing)
PacketOut

SWITCH PROGRAM

DEMO SETUP AND SCENARIOS



- **5 Gbps emulated bottleneck** in downlink direction
  - No bottleneck in uplink direction
- **TCP traffic generated by iperf3**
  - Various number of flows: 10, 40
  - Congestion control algorithm can be changed (Linux kernel support is needed)
- A **modified T4P4S P4 software switch** runs AQM algorithms described as P4 programs
  - **FIFO as reference** (basically and empty P4 program), **PIE** AQM and **RED**
  - The physical buffer size is 16384 packets ~ 24.5 Mbytes
- P4 codes at **http://lakis.web.elte.hu/aqmdemo/**

ACM SIGCOMM 2019, August 19-24, 2019, Beijing, China
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.