Active Queue Management on Tofino programmable Dataplanes

Maurice Mouw
University of Amsterdam
mmouw@os3.nl

Supervisor:
Chrysa Papagiannis
University of Amsterdam
Introduction

- Cluttered videos, audio drops or delay when playing an online game; nobody likes it.

- One of the causes for latency is the excess buffering of packets in switched networks, also referred to as bufferbloat.

- Old technologies have regained interest and new technologies are being developed to address bufferbloat.
Background

- **Active Queue Management (AQM)**; address drop-tail queues that tend to affect bursty flows and global synchronisation of flows.
  - Examples: PI2 and DualPI2.

- **Scalable Congestion Control (S-CC)**; achieve high burst tolerance, low latency and high throughput with shallow buffers.
  - Explicit Congestion Notification (ECN) marker to differentiate Classic TCP (Reno/Cubic) from S-CC.
  - Examples: DataCenter TCP (DCTCP), TCP Prague, Google’s BBRv2.

- **Low queue in Latency, Low Loss, Scalable throughput (L4S) Architecture**; improve the latency by utilizing AQM with S-CC mechanisms.¹
  - Queuing latency: less then 1 millisecond (ms) on average.
  - Less then about 2 ms at the 99th percentile.

Background Cont’d

- **P4**: domain-specific language for network devices, specifying how data plane devices called targets (switches, NICs, routers, filters, etc.) process packets.¹

- **Tofino Target and Tofino Native Architecture (TNA)**.
  - Parser/Deparser; (de)parses incoming and outgoing packets - **Programmable**
  - **Match-Action Units (MAUs)**, x12 per pipe; process packets based on configuration - **Programmable**
  - **Traffic Manager (TM)**; handles queueing of packets - **Not programmable**

¹[https://p4.org/](https://p4.org/)
²[https://docs.google.com/presentation/d/1zliBq5S81OD4mQUboRRmF_19poelleLDLaD5zLzTkVc/edit#slide=id.g37fca2850e_6_1414]
Research Question

How can the DualPI2 AQM algorithm, that can handle both scalable and non-scalable/Classic TCP variants, be implemented on a Tofino ASIC where the traffic manager is non-programmable using P4 and how does it perform?
Related Work

- De Schepper et al. implemented PI2 AQM as a Linux qdisc and ran tests using both Classic and Scalable TCP.\(^1\)

- Albisser et al. implement DualPI2 AQM as a Linux qdisc, focusing on the optimization of queue delay and window balance.\(^2\)

- Papagianni and De Schepper implemented the PI2 AQM algorithm as a P4 program using the v1model architecture for a BMv2 target.\(^3\)

\(^1\) https://doi-org.proxy.uba.uva.nl/10.1145/2999572.2999578
\(^2\) https://bobbriscoe.net/projects/latency/dualpi2_netdev0x13.pdf
\(^3\) https://doi.org/10.1145/3360468.3368199
Methods

1. Port PI2 and implement DualPI2 AQM using the TNA architecture for the Tofino ASIC.
   - Openlab emulated environment (Intel SDE 9.2.0).

2. PoC PI2 and DualPI2 AQMs on real Tofino hardware @ ELTE University in Budapest.

3. Test fairness of for PI2 and DualPI2 on the Tofino hardware.
   - Cubic (non-scalable) and Prague (scalable).
     - Validate fairness (2 flows).
   - Target delays of 20ms, 1 Gbps bandwidth, RTT of 10.

\[\text{https://doi.org/10.1145/3360468.3368189}\]
Implementation: Porting and TNA Limitations

- Can’t do multiplication or division operations.
- Only simple logical operations can be performed (if->else) in (match-)action blocks.
- Only single stage actions are allowed.
- Can’t do multiple register operations for a packet passing through the pipeline.

Algorithm 1: Psuedo code; example for alpha approximation using bit shifts.

```plaintext
error = update_laps - last_queue_delay;
constant ALPHA_1 = 10;
constant ALPHA_2 = 8;
constant ALPHA_3 = 6;
alpha_result = (error < ALPHA_1) + ( error < ALPHA_2) + ( error < ALPHA_3)
```

Example of single register access for TNA per packet, traversing the pipeline.
Implementation: Porting PI2

- Moved PI calculations to a custom control app written in C++.

1. The control plane collects **queue delay** and **timestamp** to determine probability $p'$.

2. The control plane updates the probability $p'$ every update interval $T$ and stores those in registers in the dataplane.

3. Dataplane compares $p'$ against random value $R/2$ and $R^2$ in the registers and returns boolean for comparison in the dataplane.

4. The dataplane then determines to mark or drop based on register value and ECN classification.
Results & Discussion: PI2

- Comparable results as the P4 PI2 paper on a BMv2 target using the v1 model.¹

- Queue delay is kept within the configured target of 20 ms.

- Fairness is achieved for 1 Gbps throughput using PI2.

¹ https://doi.org/10.1145/3360468.3368189
Implementation: DualPI2

1. Control plane is extended to determine $k_p'$ for $P_L$ and stores this in a register on the dataplane.

2. As there is no control over the queues in the traffic manager, (virtual) queue is created using a two rate Three Color Marker (trTCM).

3. Meter marks packets (Yellow) based on threshold $T$ defined as Committed Burst Size (CBS) and Committed Information Rate (CIR).

4. If traffic is scalable (ECT) and $k_p'$ or Yellow is True results in marking the traffic.
Results & Discussion: DualPI2

- The threshold $T$ (CBS/CIR) could not be applied on Tofino switch using the python-based API (shell) when using a C++ control plane.

- Limited documentation available on writing custom control plane programs.

- SDE source code and a found examples were used to create the control plane however this introduces a high learning curve.
Conclusions

How can the DualPI2 AQM algorithm, that can handle both scalable and non-scalable TCP variants, be implemented on a Tofino ASIC where the traffic manager is non-programmable using P4 and how does it perform?

- A complete dataplane implementation is (at this moment) not possible in Tofino, implementing both PI2 and DualPI2 is done using Control Plane program.

- The created PI2 AQM implementation provides fairness between scalable and classic TCP traffic.

- The DualPI2 implementation runs on a Tofino switch, however the configuration for the virtual queue configured via a meter needs to be set by the controlplane to do the trTCM.
Future Work

● Extent and test DualPI2 AQM implementation.

● Compare PI2 and DualPI2 AQM algorithms against other AQM algorithms created for TNA (e.g. PIE, FQ-CODEL and/or VDQ-CSAQM).

● Rework the code to be more (resource) efficient.

● Tests if the newer 9.5.0 SDE allows more operations in the dataplane.