A Congestion Control Independent L4S Scheduler

Szilveszter Nádas*, Gergő Gombos†, Ferenc Fejes†, Sándor Laki†

* Ericsson Research, Budapest, Hungary
† ELTE Eötvös Loránd University, Budapest, Hungary

Contact: lakis@inf.elte.hu  Web: http://ppv.elte.hu
Low latency is important for many applications

• Not only for traditional **non-queue-building traffic**
  • DNS, gaming, voice, SSH, ACKs, HTTP requests, etc.

• But for **throughput hungry applications** as well
  • HD/4K or holographic video conferencing, AR/VR, remote control/presence, cloud-rendered gaming, etc.

• Simple strict priority scheduling is not enough
How to ensure low latency and high throughput?

• Affected by **both end-systems and the network**
  • E.g., congestion control (CC), queue management (QM)

• **Classic TCP CC needs large queues** to achieve full link-utilization
  • Filling the buffers by design - large buffering delay
  • With AQM the latency is still too large (~RTT)

• **Scalable CC** (e.g., DCTCP, BBRv2, Prague) **ensures ultra-low latency**
  • Tiny buffers are enough for full utilization, but ECN support is needed
  • Too aggressive for the coexistence with Classic TCP
L4S = Low Latency, Low Loss & Scalable Throughput

- L4S promises **ultra-low queuing delay over the public Internet**

- Design goals of an L4S AQM
  - **Isolation** of L4S service from Classic
  - **Coexistence** between L4S and Classic flows

- Current „state-of-the-art“ proposal
  - DualQ AQM – **DualPI2 AQM**

*Source: O. Albisser et al. „DUALPI2 - Low Latency, Low Loss and Scalable (L4S) AQM”, in Proc. Netdev 0x13 (Mar 2019).*
State-of-the-art proposal DualPI2

- Different congestion signal intensity for L4S and Classic queues
- Low latency
- Window fairness

Native L4S AQM
STEP (or RED) AQM
ECN marking

The two AQMs are coupled.
(Higher signal probability for L4S, lower for Classic.)

Source: O. Albisser et al. „DUALPI2 - Low Latency, Low Loss and Scalable (L4S) AQM”, in Proc. Netdev 0x13 (Mar 2019).
Are we done?

• **Separation** of Classic and Scalable traffic
  • Assuming a single Classic and Scalable CC behavior

• **Different Classic and Scalable CC proposals**

• **Incompatible CCs** inside the same CC family
  • Different CCs and/or different RTTs
  • Classic CCs - **Cubic is more aggressive than Reno**, there are **RTT unfairness**, etc.
  • Scalable CCs - **Are the scalable mechanisms of BBRv2 and DCTCP compatible?**

• AQM compatibility?
DCTCP vs. BBRv2, 1 Gbps, 5 ms RTT

Typically DC wins for STEP

Reasonable fairness

Using in-network resource sharing

Source: F. Fejes et al. „On the Incompatibility of Scalable Congestion Controls over the Internet“, FIT WS@IFIP Networking 2020
**DCTCP vs. BBRv2, 1 Gbps, 5 ms RTT**

- DCTCP and BBRv2 require different signal intensities
- STEP AQM applies the same ECN marking probability
- Leading to unfairness

---

*Source: F. Fejes et al. „On the Incompatibility of Scalable Congestion Controls over the Internet”, FIT WS@IFIP Networking 2020*
DCTCP vs. BBRv2, 1 Gbps, 5 ms RTT

- CSAQM can provide **different signal probabilities**
  - without flow identification or per-flow queues
- BUT **cannot satisfy the requirements of L4S and Classic traffic** at the same time
- Requires additional packet marking before the bottleneck
  - Incentive used for deciding on forward or drop/ECN-mark a packet

**Source:** F. Fejes et al. „On the Incompatibility of Scalable Congestion Controls over the Internet”, FIT WS@IFIP Networking 2020
Our approach is based on the **Per Packet Value** framework

**Packet Marker** at the edge of the network
- Stateful, but highly *distributed*
- Assigning values to packets
- Packet values are *incentives* helping to decide which packet to forward/drop in case of congestion

**Resource Nodes** (e.g. routers) aim at maximizing the total transmitted Packet Value.
- Stateless and *simple*
- Drop packets with *minimum value first strategy* if packet arrives at a full buffer
Throughput (Mbps)

Packet Value

Flow #1

Flow #2

Creating a BN

CNL

BN

100 Mbps

Creating

Sending rate $R_1 = 80\text{Mbps}$

Resource share at BN $th_1 = 30\text{Mbps}$

$th_2 = 30\text{Mbps}$

Congestion

$CTV = 8$
Our L4S AQM algorithm
Virtual DualQ Core-Stateless AQM (VDQ-CSAQM)
Our L4S AQM algorithm
Virtual DualQ Core-Stateless AQM (VDQ-CSAQM)

- **Two physical queues**
  - Separating L4S and Classic tr.

- **Two virtual queues (VQs)**
  - VQ$_0$ for L4S traffic only
  - VQ$_1$ for both L4S and Classic

- **Each VQ**
  - only stores meta-information (PV and packet size)
  - has a **max. size** and a serving rate $C_{vi} \leq C$
  - has a **PV histogram** reflecting the PV distribution in the VQ
Our L4S AQM algorithm
Virtual DualQ Core-Stateless AQM (VDQ-CSAQM)

- **Strict priority scheduler**
  - Simple and available in HW switches

- **CTV \_i calculated from**
  - PV histogram of VQ\_i, H\_IN\_i
  - Delay target D\_i
  - Periodically (every 10 ms)

- **Dequeue from L4S queue (Queue 0)**
  - If PV > max (CTV\_0, CTV\_1), forward
  - Else mark packet with CE
  - Update both VQs and histograms

- **Dequeue from Classic queue (Queue 1)**
  - If PV > CTV\_1, forward the packet
  - Else drop (or ECN mark) the packet
  - Update VQ\_1 and its histogram
Evaluation

Testbed setup

- Intel Xeon 6 core CPU (3.2GHz)
- TCP traffic generated with iperf2
  - Flows start at the same time
- BBRv2 alpha kernel (5.4.0-rc6)
  - Default settings: no pacing for DCTCP, internal pacing of BBRv2
- ACKs are delayed to emulate propagation RTT
- AQMs implemented in DPDK
  - DualPI2 is based on „draft-ietf-tsvwg-aqm-dualq-coupled-11“
Dynamic traffic – equal RTT (5ms)

DCTCP – Cubic CCs

VDQ-CSAQM

DualPI2

#L4S-Cl. flows 1-0 1-1 10-1 10-10 50-10 50-50 10-50 1-10 0-1

Class Thr. [Mbps]

Flow Thr. [Mbps]

Average Delay [ms]
Dynamic traffic – equal RTT (5ms)

DCTCP – Cubic CCs

**VDQ-CSAQM**

- Class Thr. [Mbps]
- Flow Thr. [Mbps]
- Average Delay [ms]

- #L4S-Cl. flows: 1-0, 1-1, 10-1, 10-10, 50-10, 50-50, 10-50, 1-10, 0-1

**DualPI2**

- Class Thr. [Mbps]
- Flow Thr. [Mbps]
- Average Delay [ms]

- #L4S-Cl. flows: 1-0, 1-1, 10-1, 10-10, 50-10, 50-50, 10-50, 1-10, 0-1

Good flow fairness if the number of flows is large.
Dynamic traffic – equal RTT (5ms)

DCTCP – Cubic CCs

VDQ-CSAQ M

DualPI2

VQs lead to underutilization by design
Dynamic traffic – equal RTT (5ms)

DCTCP – Cubic CCs

VDQ-CSAQM

DualPI2

Low utilization with a single DCTCP flow

No such problem with a single Classic flow
Dynamic traffic – equal RTT (5ms)

DCTCP – Cubic CCs

VDQ-CSAQM

1 L4S and 1 Classic flows - significant unfairness
Dynamic traffic – equal RTT (5 ms)

DCTCP – Cubic CCs

VDQ-CSAQM

DualPI2

#L4S-Cl. flows

#L4S-Cl. flows

Class Thr. [Mbps]

Class Thr. [Mbps]

Flow Thr. [Mbps]

Flow Thr. [Mbps]

Average Delay [ms]

Average Delay [ms]
Dynamic traffic – equal RTT (5ms)
BBRv2 – Cubic CCs

**VDQ-CSAQM**

- Class Thr. [Mbps]
- Flow Thr. [Mbps]
- Average Delay [ms]

**DualPI2**

- Class Thr. [Mbps]
- Flow Thr. [Mbps]
- Average Delay [ms]
Dynamic traffic – equal RTT (5ms)

**BBRv2 – Cubic CCs**

**VDQ-CSAQM**

BBRv2 applies a model-based CC, but what if the network works with a different model.

**DualPI2**

BBRv2 L4S flows dominate, surprising Classic ones.
Dynamic traffic – equal RTT (5ms)

BBRv2 – Cubic CCs

**VDQ-CSAQM**

Worst fairness
7:3 L4S:Classic ratio
Dynamic traffic – equal RTT (5ms)
BBRv2 – Cubic CCs

VDQ-CSAQM

DualPI2
Heterogeneous RTT (5ms and 40ms)

#Flows (L4S-5ms, L4S-40ms, Cl-5ms, Cl-40ms)

- **DCTCP w. 5ms RTT gets higher share**

**DCTCP - Cubic**

**BBRv2 - Cubic**

![Flow Thr. [Mbps] vs Time [s] for different protocols and flows](image1)

**VDQ-CSAQM**

**DualPI2**
Heterogeneous RTT (5ms and 40ms)

#Flows (L4S-5ms, L4S-40ms, Cl-5ms, Cl-40ms)

**DCTCP w. 5ms RTT gets higher share**

**DCTCP - Cubic**

**BBRv2 - Cubic**
Heterogeneous CCs and equal RTT (5ms)

L4S: **DCTCP** & **BBRv2 (ECN)** – Classic: **Cubic** & **BBRv2 (drop)**

#Flows (L4S-DC, L4S-BBR, Cl-Cubic, Cl-BBR)

![Graph showing flow throughput over time for different configurations.](image1)

**VDQ-CSAQM**

![Graph showing flow throughput over time for different configurations.](image2)

**DualPI2**
Heterogeneous CCs and equal RTT (5ms)

L4S: DCTCP & BBRv2 (ECN) – Classic: Cubic & BBRv2 (drop)

#Flows (L4S-DC, L4S-BBR, Cl-Cubic, Cl-BBR)

VDQ-CSAQM

DualPI2
Heterogeneous CCs and equal RTT (5ms)
L4S: DCTCP & BBRv2 (ECN) – Classic: Cubic & BBRv2 (drop)

#Flows (L4S-DC, L4S-BBR, Cl-Cubic, Cl-BBR)
Conclusion

• **CC evolution is ongoing**
  • Compatibility of CCs even within the same CC family (either classic or scalable) **cannot be expected**

• **Different congestion signal intensities within the same CC family**
  • Flow identification or **additional incentives like packet value**

• VDQ-CSAQM works well with heterogeneous CCs and RTTs
  • supports the **coexistence of even incompatible congestion controls**
  • provides **ultra-low latency for L4S flows**
  • while **keeping the bottleneck utilization reasonable (98.4% caused by VQs)**.

• VDQ-CSAQM can provide **different signal intensities for various flows**
  • **Without flow identification and per-flow queueing**

• We also work on the P4 implementation of VDQ-CSAQM

• **All the measurement results (incl. ones at 10 Gbps) are available**
  • [http://ppv.elte.hu/cc-independent-l4s/](http://ppv.elte.hu/cc-independent-l4s/)