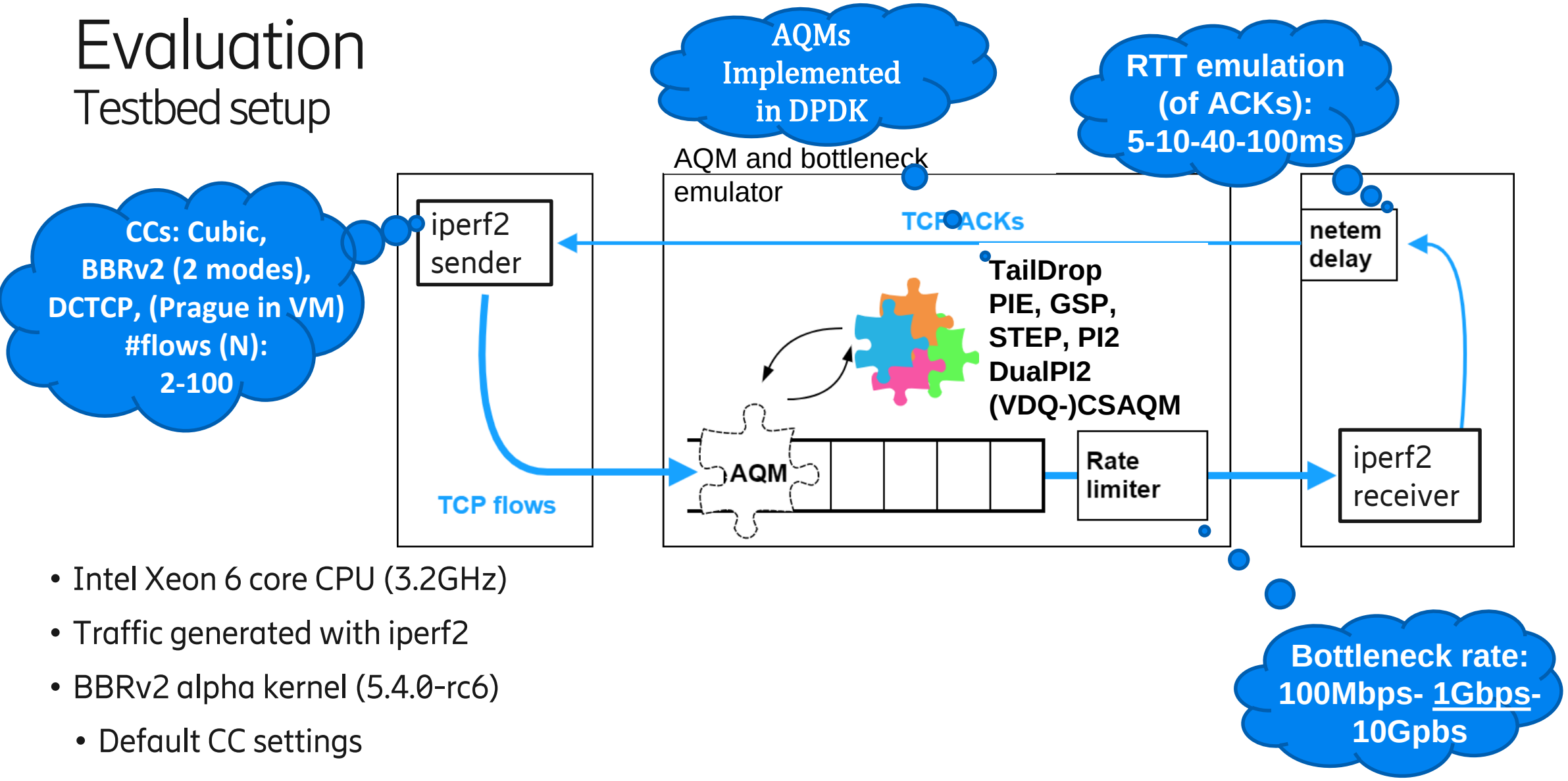
On Congestion Control (un)fairness

Szilveszter Nádas, Balázs Varga (Ericsson Research, Hungary)
Ferenc Fejes, Gergő Gombos, Sándor Laki (ELTE Eötvös Loránd University, Hungary)

Evaluation
Testbed setup

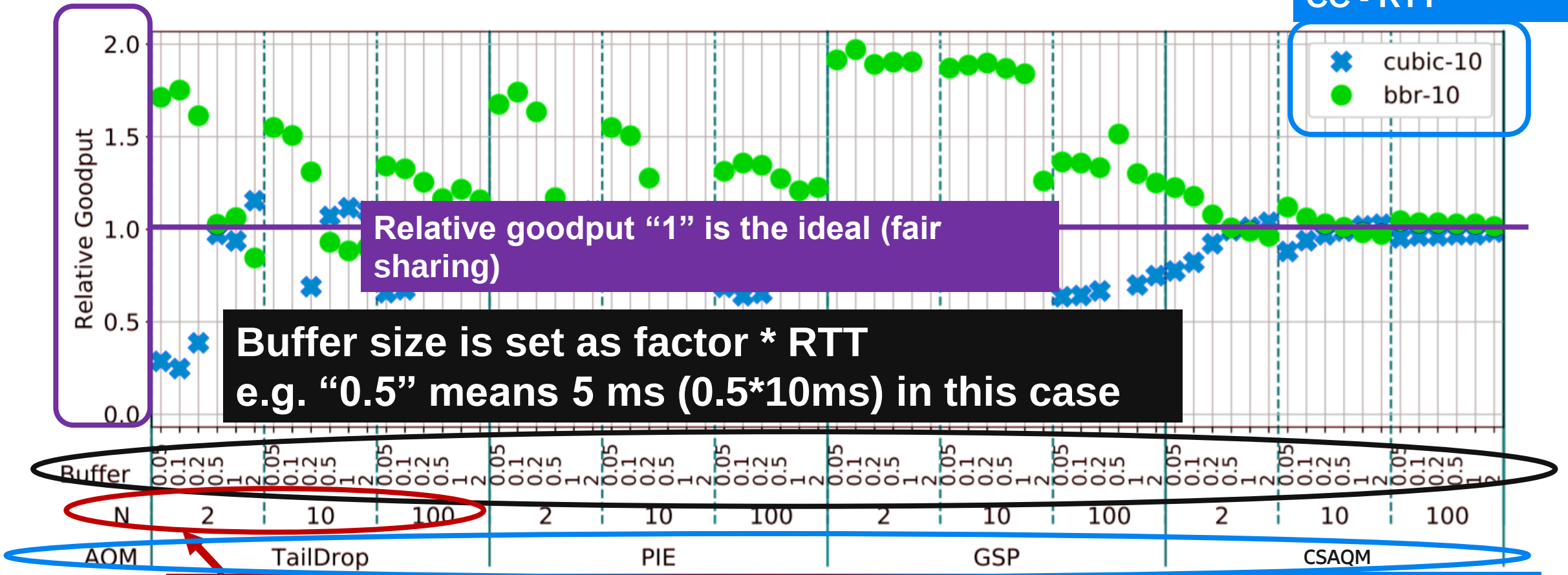


- Intel Xeon 6 core CPU (3.2GHz)
- Traffic generated with iperf2
- BBRv2 alpha kernel (5.4.0-rc6)
  - Default CC settings
- ACKs are delayed to emulate propagation RTT (there are multi-RTT scenarios)
- AQMs implemented in DPDK

Cubic vs BBR2, 1Gbps, 10ms RTT

Relative goodput of a connection class
- **Connection class: same RTT, same CC**
- **Average goodput (within the connection class) / "ideal per connection fair share"**

Connection classes
CC - RTT



Relative goodput "1" is the ideal (fair sharing)

Buffer size is set as factor * RTT
e.g. "0.5" means 5 ms (0.5*10ms) in this case

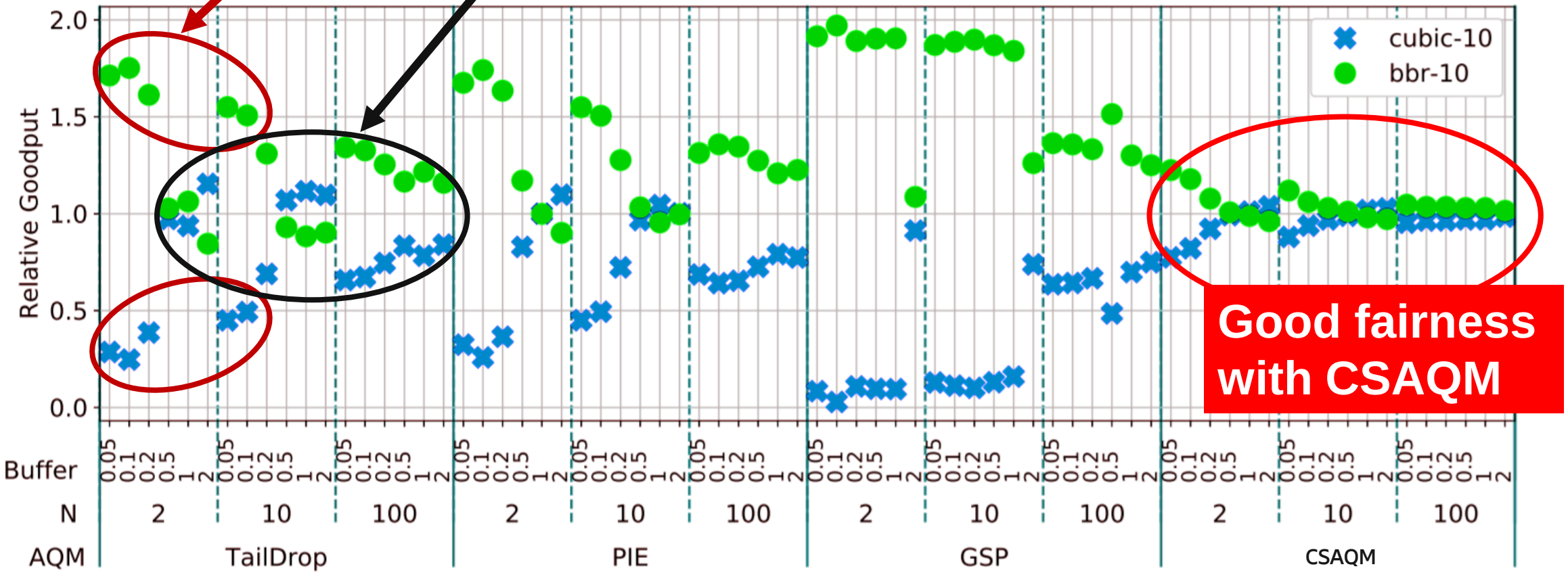
N: Number of connections
Half is from a connection class (N=10 → 5 Cubic + 5 BBR)

Studied AQMs

Cubic vs BBR2, 1Gbps, 10ms RTT

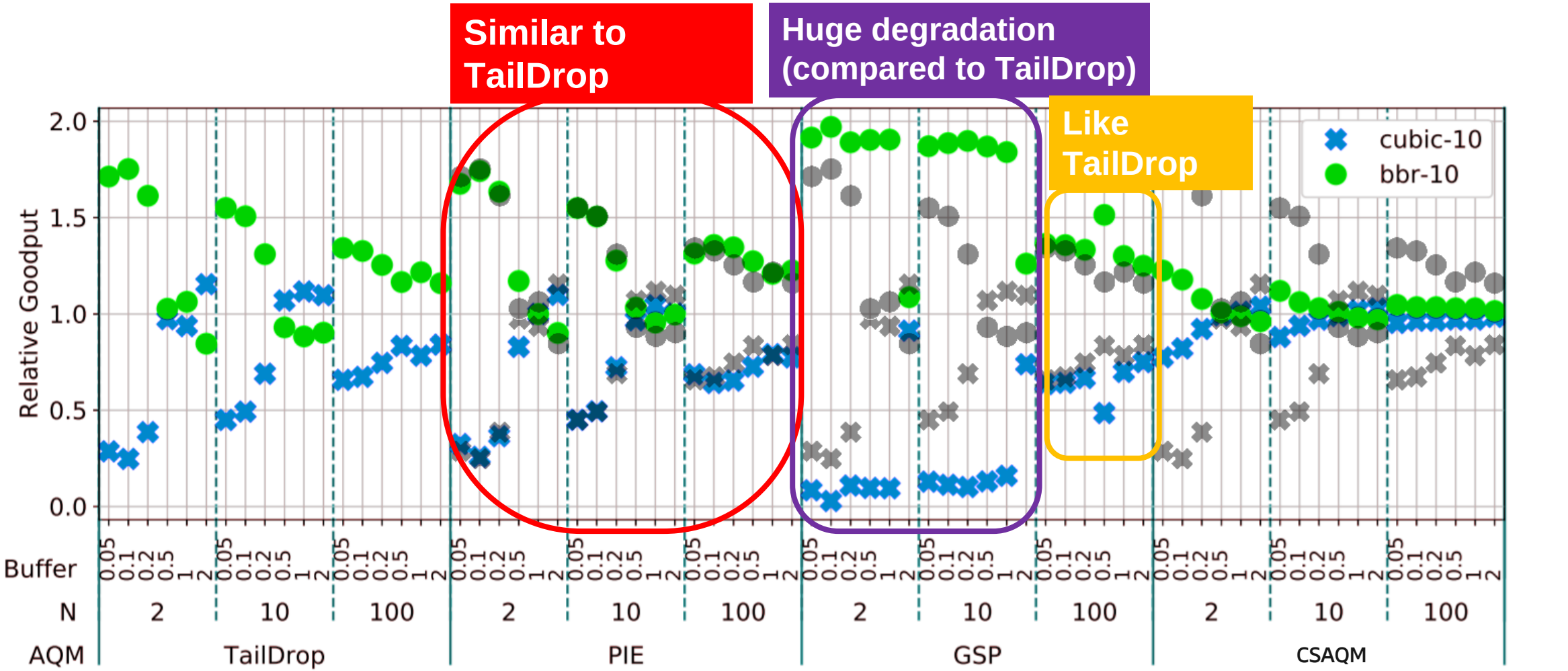
Worse for Small buffers

Reasonable fairness



Good fairness with CSAQM

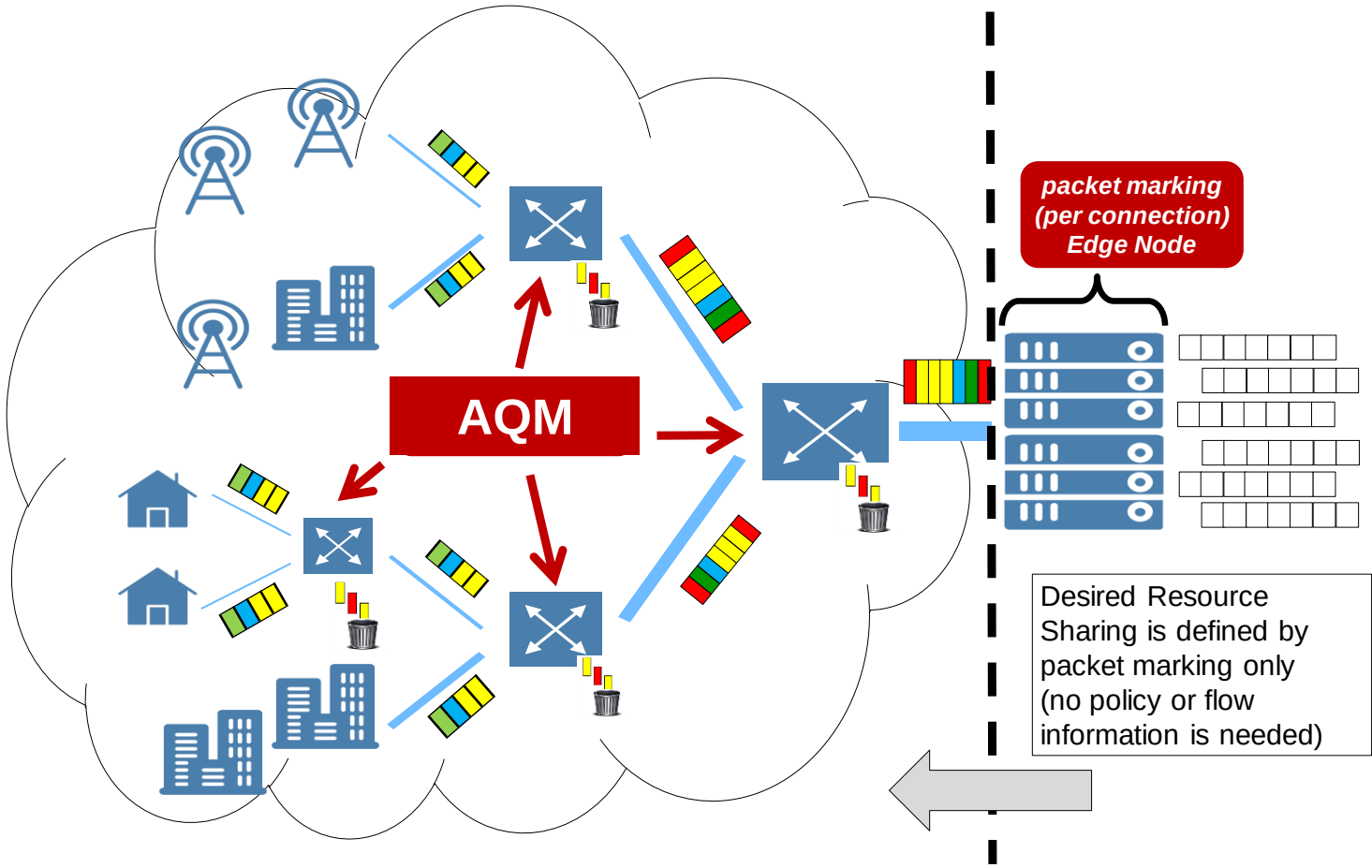
Cubic vs BBR2, 1Gbps, 10ms RTT



Gray shadow: TailDrop (for reference)

CSAQM/VDQ-CSAQM (Virtual Dual Queue -) Core Stateless AQM

- CSAQM is a Core-Stateless Resource Sharing framework, which
  - allows a wide variety of detailed and flexible policies;
  - enforces those policies for all traffic mixes; and
  - scales well with the number of flows
- Packet Marking at the edge (or at the end)
  - flows (or traffic aggregates) have to be identified
  - encodes policy into a value marked on each packet
    - packet header field needed
- Resource Node – AQM
  - behavior based on packet marking only
  - no need for
    - policy information
    - flow identification or flow state
    - separate queues per flow
  - very fast and simple implementations exist (P4 Tofino)



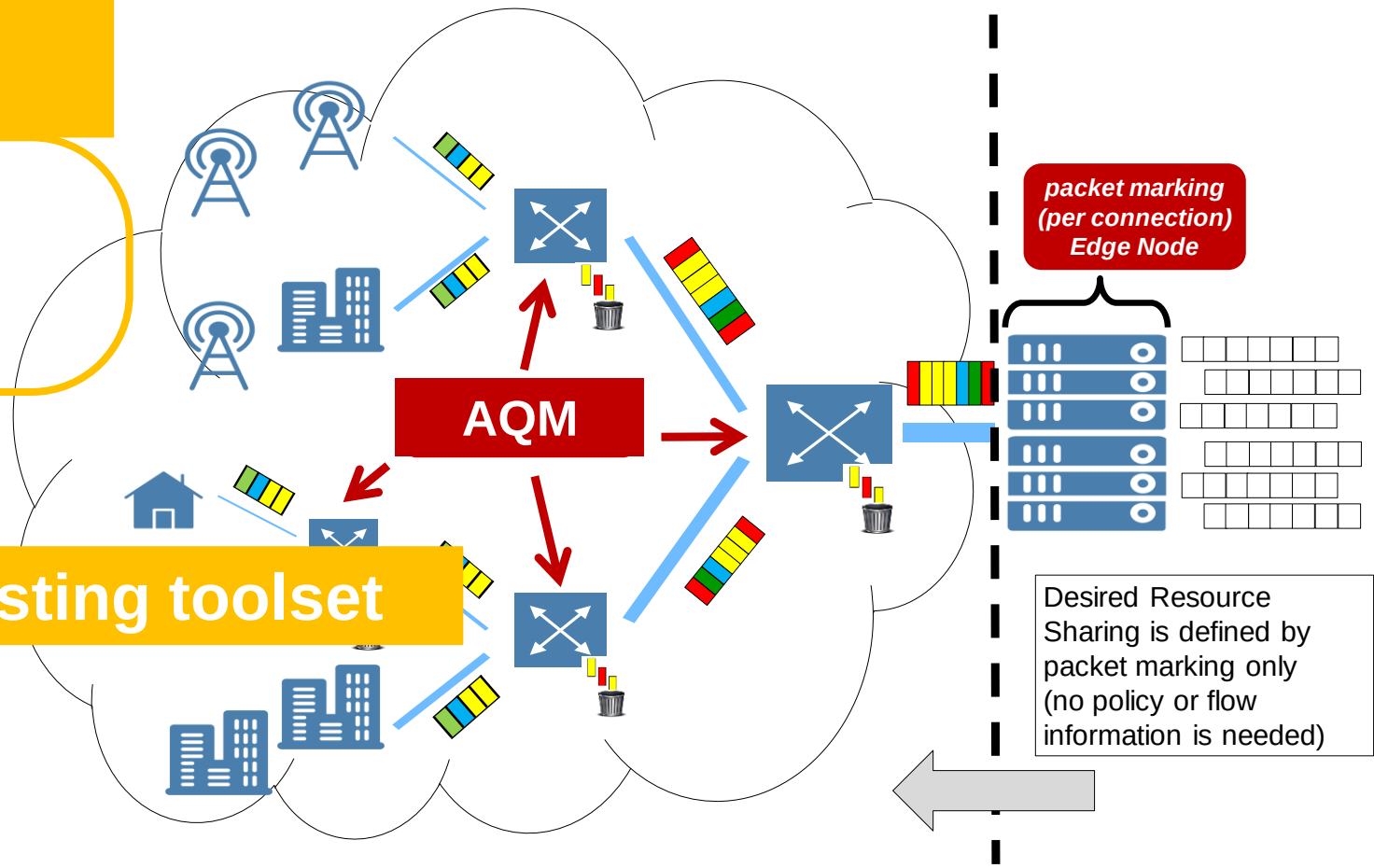
CSAQM/VDQ-CS Tutorial video @ ppv.elte.hu
(Virtual Dual Queue -) Core Stateless AQM

- CSAQM is a Core-Stateless Resource Sharing framework, which

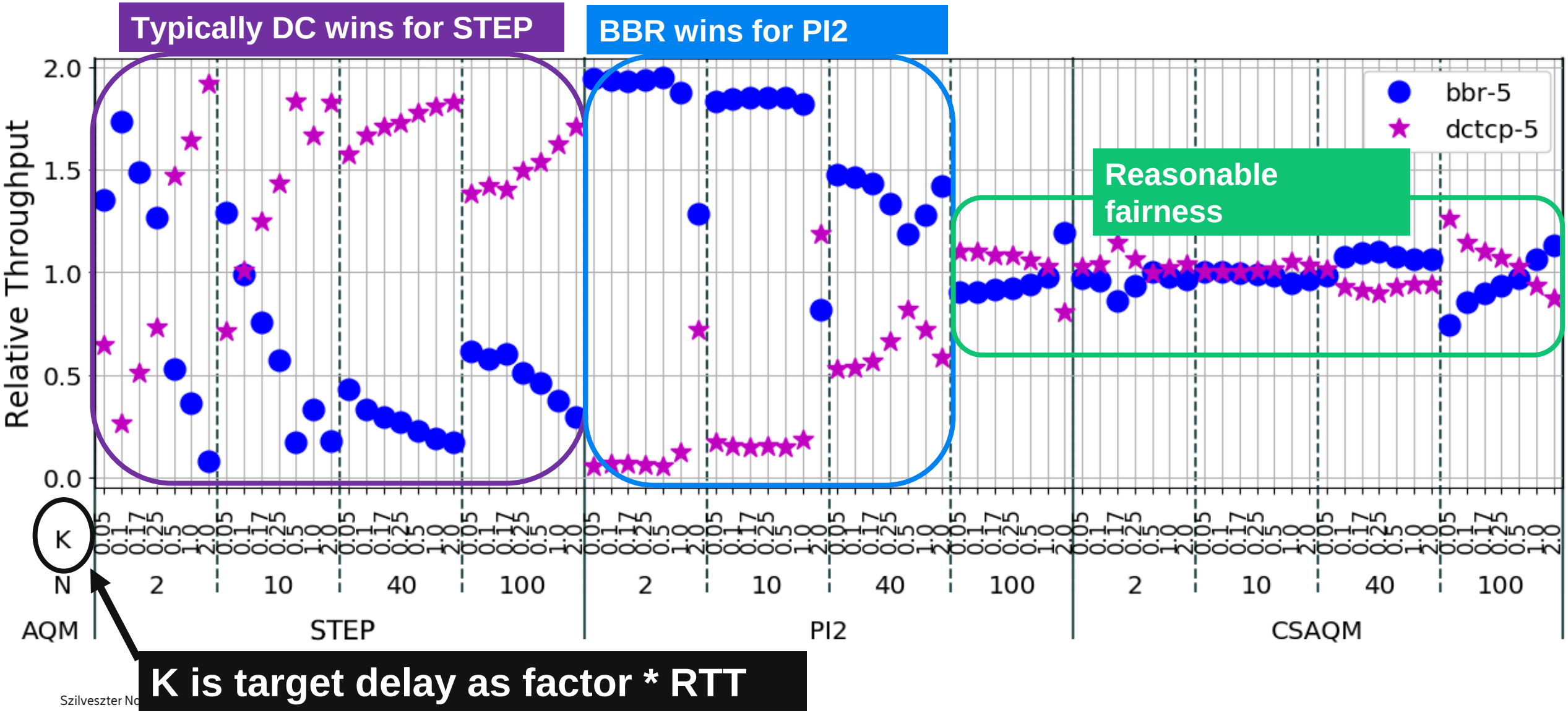
Needs standardization / within admin domain

- Packet Marking at the edge (or at the end)
  - flows (or traffic aggregates) have to be identified
  - encodes policy into a value marked on each packet
    - packet header field needed
- Resource Node – AQM
  - behavior based on packet marking only
  - no need for
    - policy information
    - flow identification or flow state
    - separate queues per flow
  - very fast and simple implementations exist (P4 Tofino)

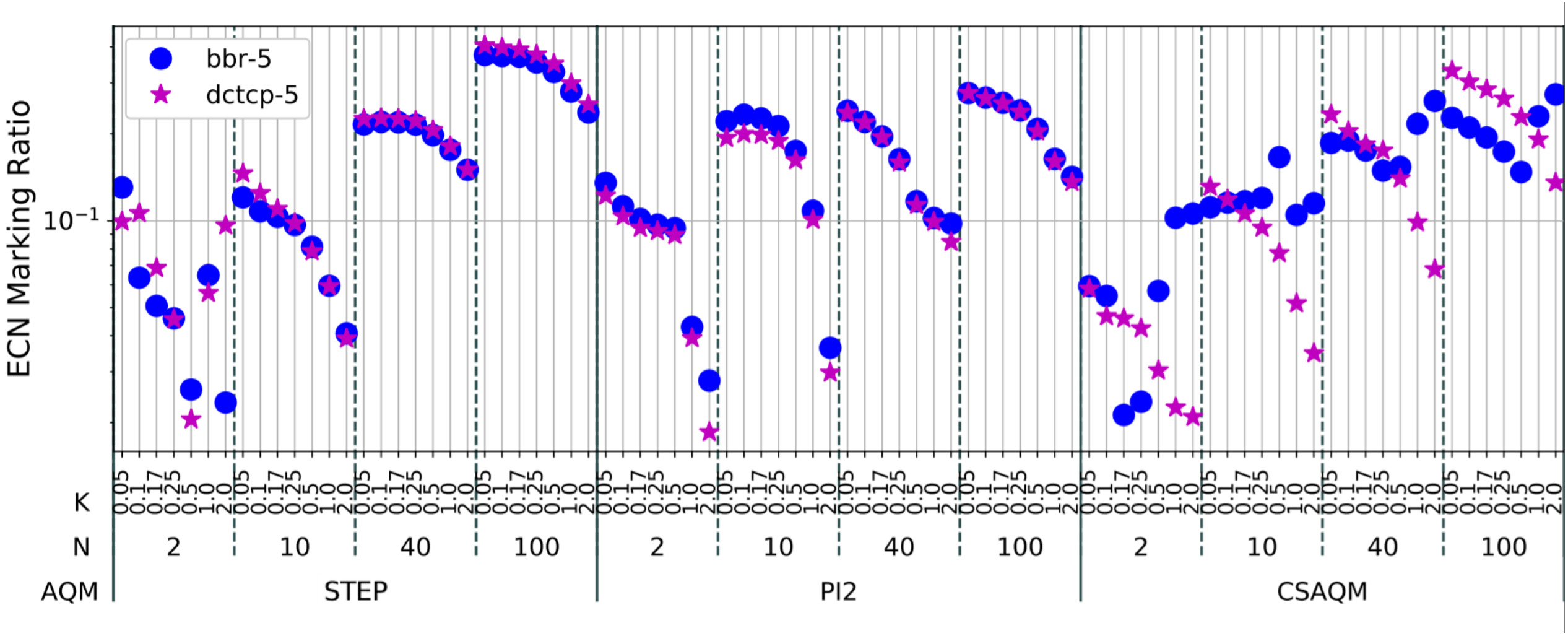
Almost existing toolset



DC vs. BBRv2, 1 Gbps, 5 ms RTT



BBRv2 vs. DCTCP: ECN marking ratio

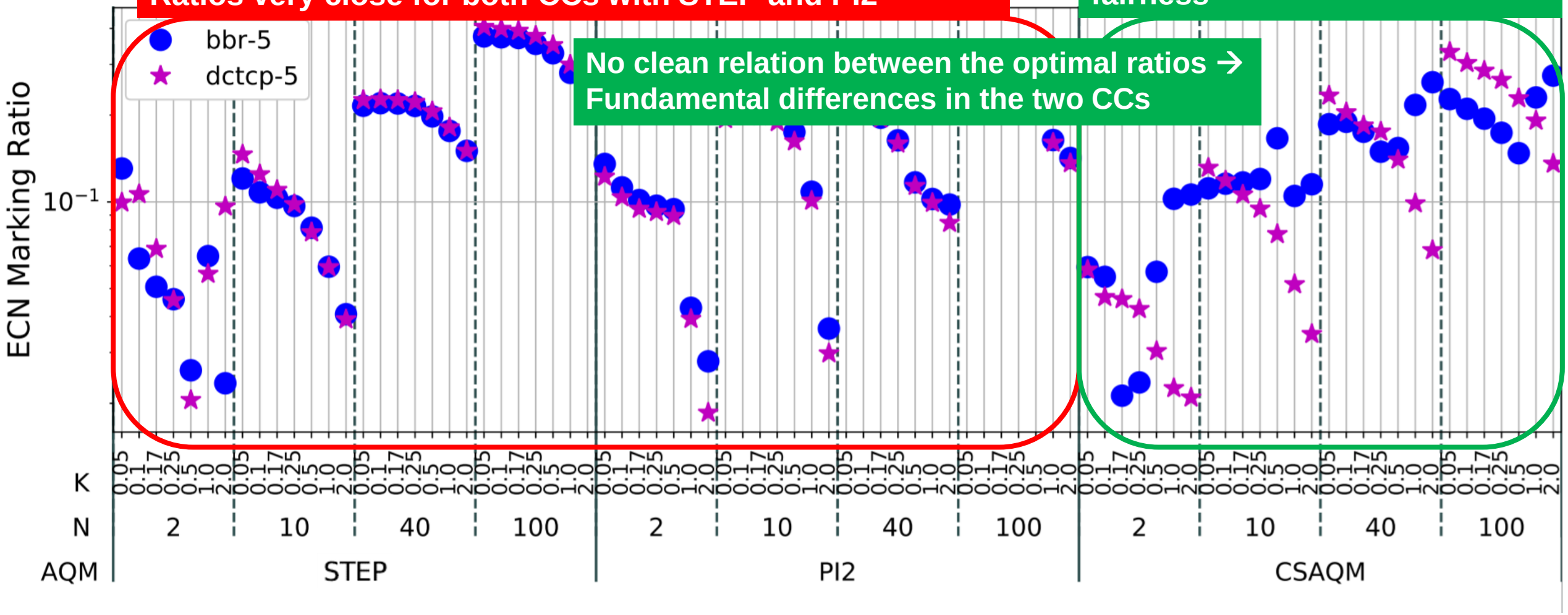


BBRv2 vs. DCTCP: ECN marking ratio

Ratios very close for both CCs with STEP and PI2

CSAQM finds the right marking ratio for the CCs to achieve fairness

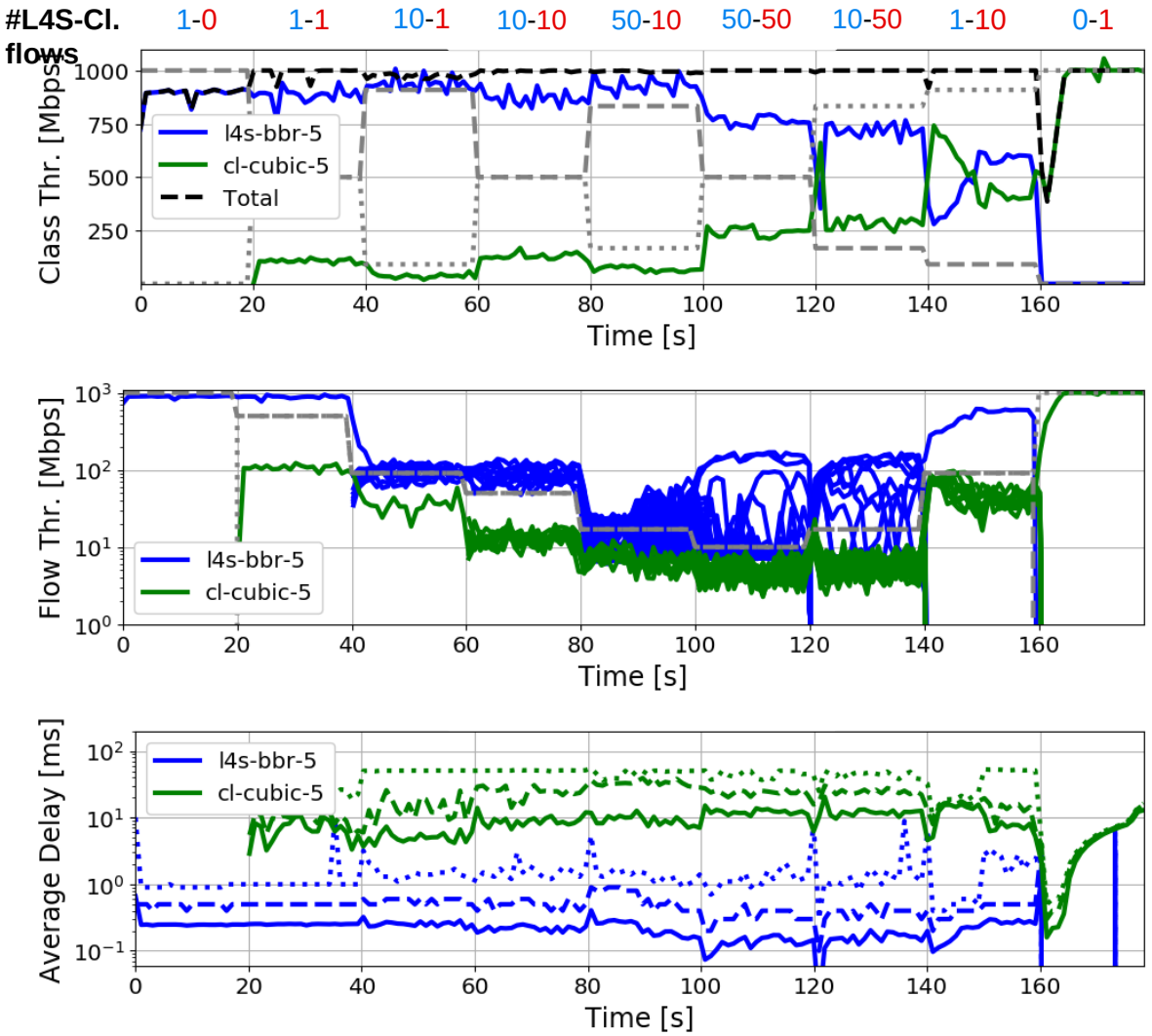
No clean relation between the optimal ratios → Fundamental differences in the two CCs



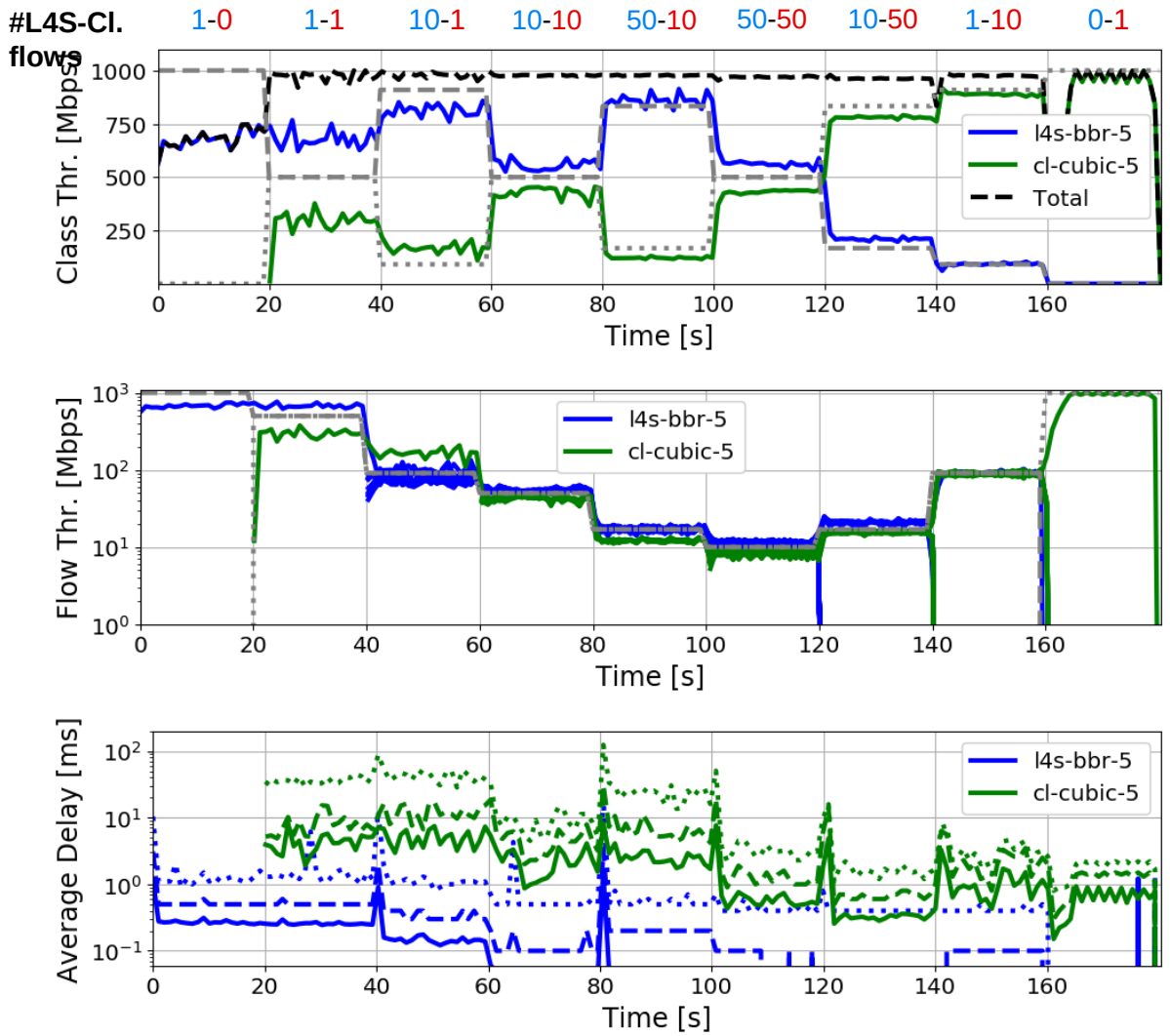
Dynamic traffic – equal RTT (5ms)

BBRv2 (scalable) – Cubic CCs

DualPI2



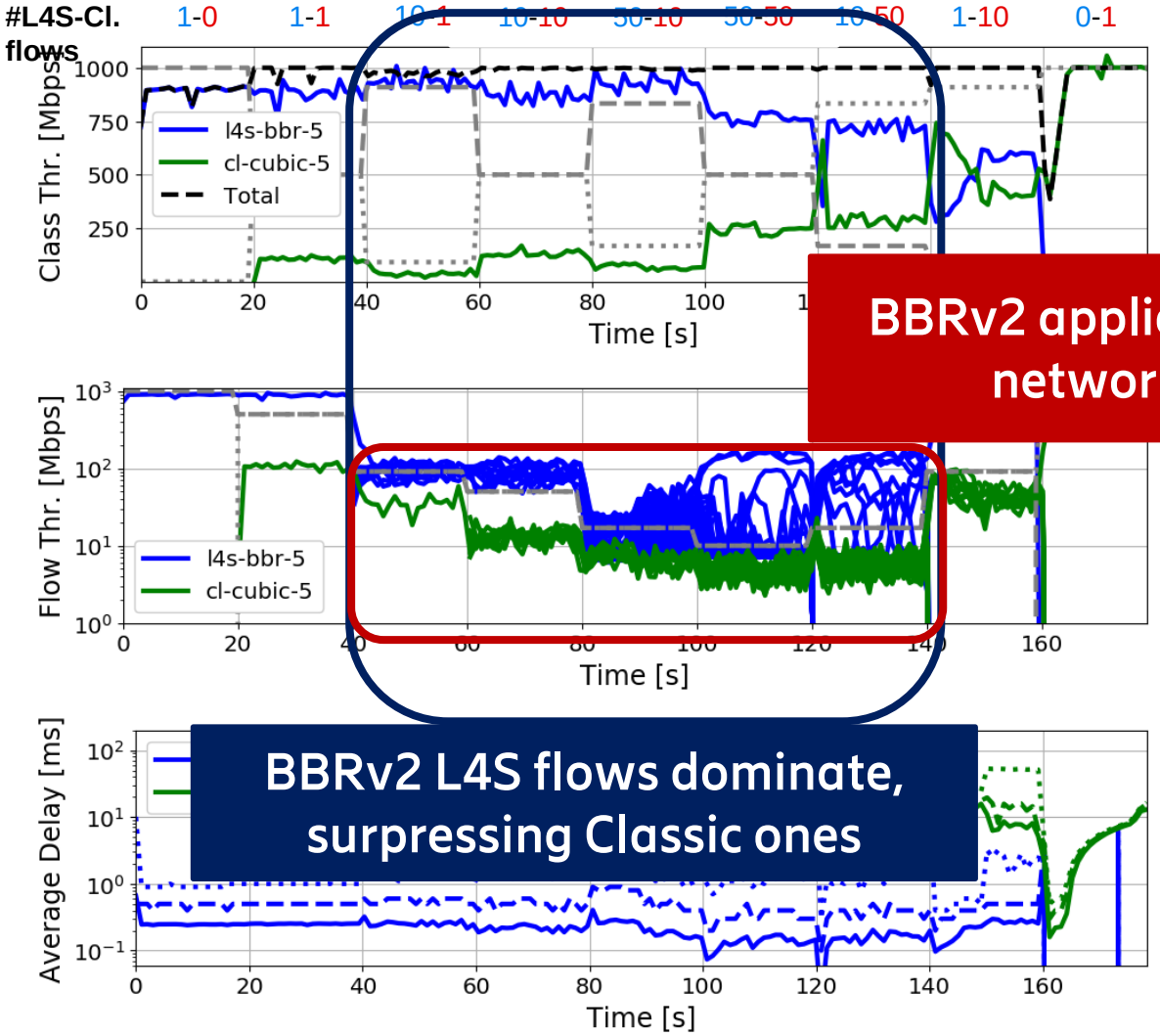
VDQ-CSAQM



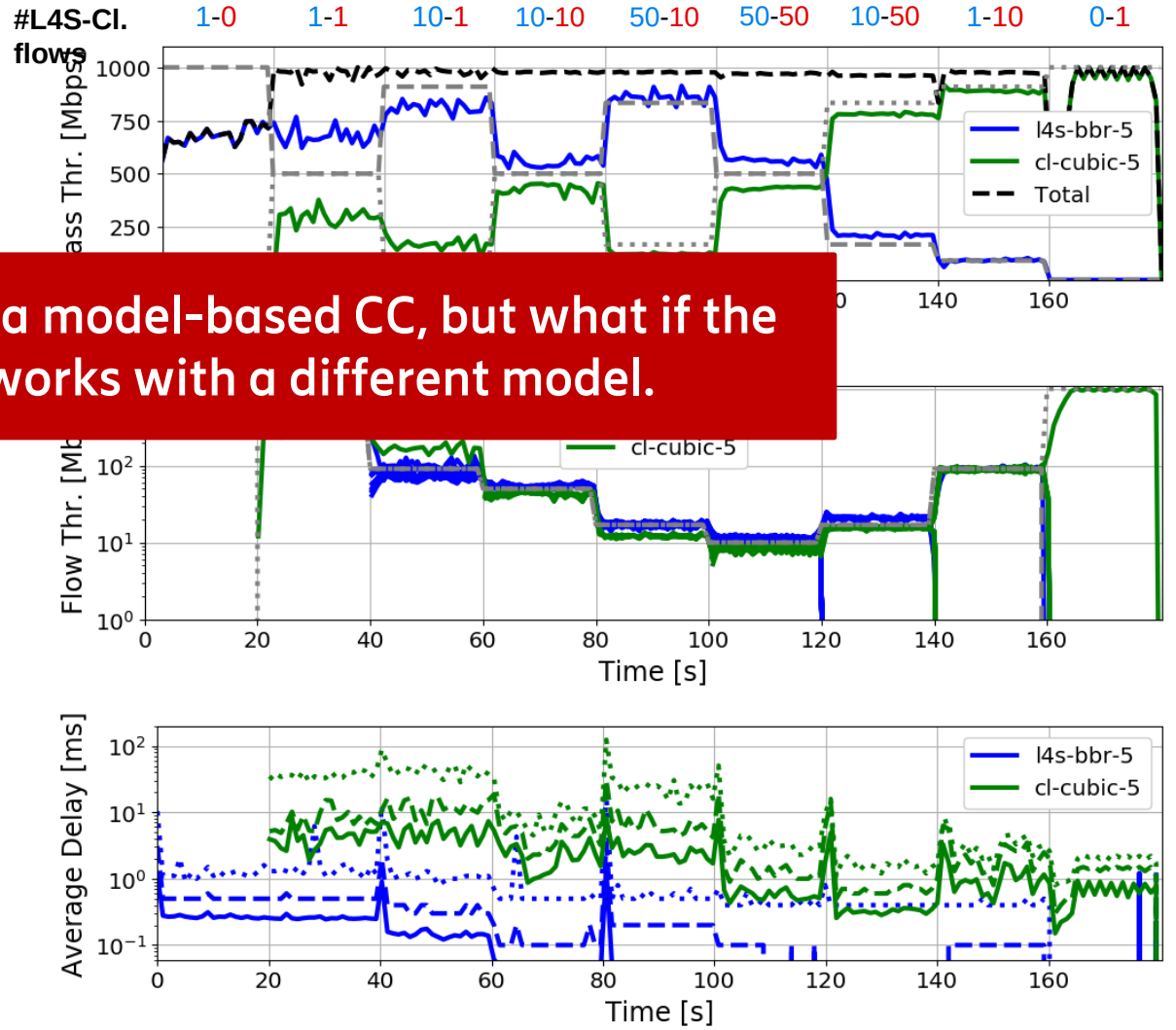
Dynamic traffic – equal RTT (5ms)

BBRv2 (scalable) – Cubic CCs

DualPI2



VDQ-CSAQM



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

BBRv2 L4S flows dominate, surpressing Classic ones

Summary of Testbed measurements

- Most CCs have RTT fairness issues even in mono-CC scenarios
- Evolved Congestion Controls have fairness issues with legacy
  - BBRv2 vs. Cubic fairness is very dependent on settings, sometimes good, sometime quite bad
  - DCTCP vs. BBRv2 (Scalable mode) in general bad fairness
- AQMs tuned for a specific CC have the potential to hurt the coexistence even more, very rarely help it
  - Even though they help e.g. multi RTT fairness when the specific CC is used
  - Examples for degraded performance (compared to TailDrop or STEP):
    - PIE and GSP for BBRv2 vs. Cubic
    - PI2 for DCTCP vs. BBRv2 (Scalable mode)
    - DualPI2: BBRv2 (Scalable mode) vs. Cubic

Summary: Congestion Control Evolution

- The Congestion Control evolution has accelerated
  - User space CC in QUIC, CC in BPF since Linux 5.6, pacing accelerated with NIC, etc.

- **It is very hard for a new CC to be both innovative and to be fair to existing CCs**
  - *"TCP-friendliness greatly constrains how we handle congestion in the Internet"* [1]
  - Fairness to existing CCs is often demonstrated in special cases, but that is not universal
  - As the number of deployed CCs increases, it is even harder
  - Even the Harm based "bar" for a new CC is close to impossible to meet (watch IRTF open meeting [2])

- Two specific CCs e.g. Prague and BBRv2-Scalable might be possible to tune to be compatible for some scenarios
  - We are skeptical that this can be generic for different RTTs, AQMs and traffic mixes
  - Or among several new CCs

How to provide fairness
Ways forward

- **Fairness by E2E CC + Overprovisioning**
  - Is it sill “the way”? Or is TCP friendliness (to Reno and/or DC) a point of ossification?

- **Fairness by scheduling in network (e.g. fq-Your Favorite AQM and HQoS) has its own issues**
  - Per flow and hierarchical queueing is not practical for high speed routers
  - **Equal (or even static) sharing is not always optimal [4]**
  - Communicating policies to every potential bottleneck node is hard

- **We believe that cooperative approaches like CSAQM has a good potential for controlling resource sharing**
  - Flow identification and policy decisions at the endpoints or at the network edge
  - CSAQM implementation in the routers is very simple and invariant to the number of flows or policies
  - Though it requires a header field

Comparison of methods providing fairness over the Internet

Methods	E2E CC	In-Network (fq-*, HQoS)	Cooperative
Fairness by	CC	Scheduling	Marking + AQM
Fairness	Has issues	Very good	Good
Resource sharing is	Dynamic	Static	Dynamic
End-host control	Full	Limited	High (endpoint marking)/ Limited (edge marking)
CC evolution	Constrained (by harm to existing CCs)	Less constrained (if every flow has separate queues)	Less constrained
Bottleneck complexity	Low	High (CPU)	Medium (P4)
Signaling complexity	n.a.	High (every potential bottleneck)	Low (endpoint marking)/ Medium (edge marking)
Standardization	no (TCP friendliness)	Signaling	Packet marking
Delay differentiation	L4S	By separate queues	L4S
RTT unfairness	Hard to solve	Solved	Solved

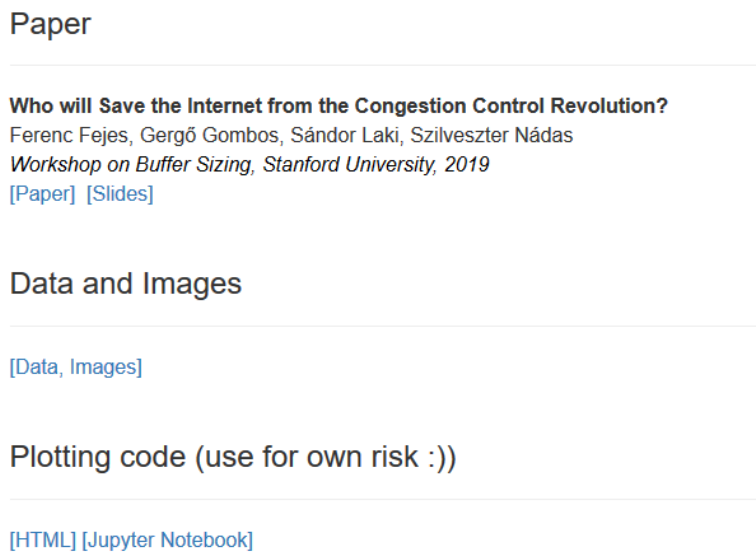
Questions to community, future work

- What more to include in these type of evaluations?
  - CCs, AQMs, RTTs, traffic patterns?
  - We use Ubuntu and BBR alpha kernel defaults,
    - Are there more meaningful defaults? (we will look into the effect of TCP HyStart)
    - More typical OS?
- Where are the typical bottlenecks?
  - What is the speed of them?
  - How many bottlenecks to consider?
- What is the effect of sub-millisecond Internet [3] on fairness?
  - Caches are very close to edge – do non-CDN flows still have a chance?

Much more results at

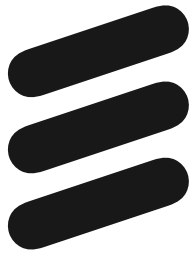
Articles, presentation material, videos, and detailed results

- http://ppv.elte.hu/buffer-sizing/ (BBRv2 vs. Cubic)
- http://ppv.elte.hu/scalable-cc-comp/ (DCTCP vs. BBRv2)
- http://ppv.elte.hu/cc-independent-l4s/ (L4S, including all of the above)
- http://ppv.elte.hu/tcp-prague/ (preliminary, L4S, TCP Prague instead of DCTCP)



References

1. Brown, Lloyd, et al. "On the Future of Congestion Control for the Public Internet." *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 2020.
2. Ware, Ranysha, et al. "Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms." *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. 2019.
3. Trevisan, Martino, et al. "Five years at the edge: Watching internet from the isp network." *IEEE/ACM Transactions on Networking* 28.2 (2020): 561-574.
4. Briscoe, Bob. *Per-Flow Scheduling and the End-to-End Argument*. Discussion Paper TR-BB-2019-001, bobbriscoe.net, 2019.



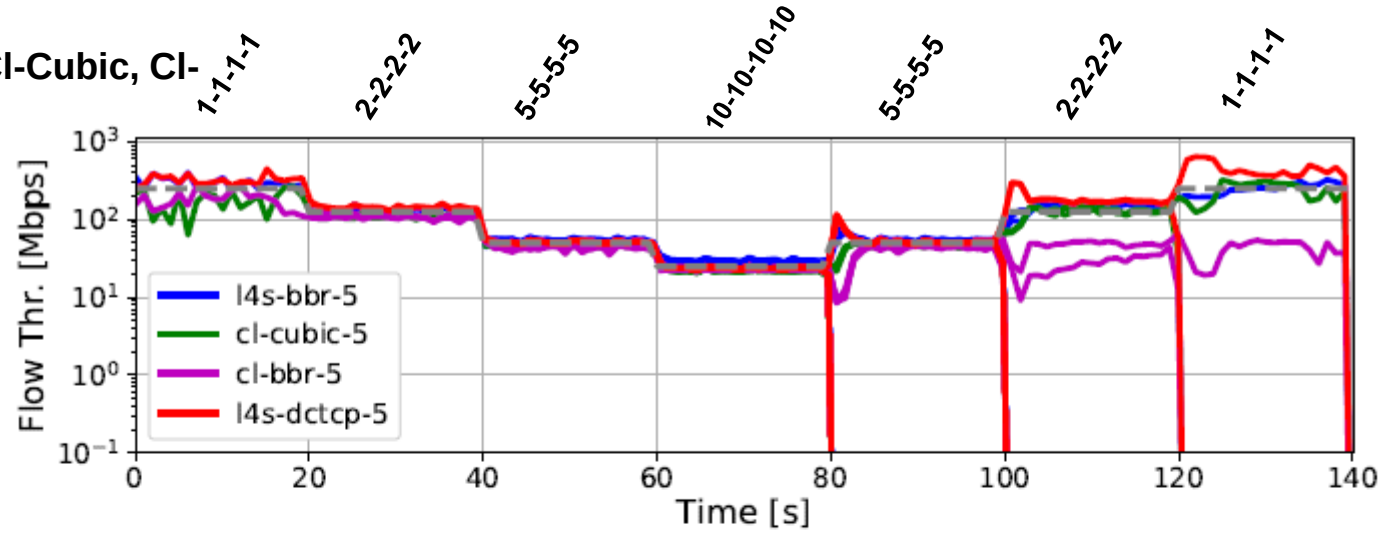
Backup slide
On per flow scheduling including fq-* AQMs

- Resource Sharing is probably even better with a well chosen fq- AQM
- Delay is likely better with VQ AQMs
  - Though VQ fq-* might be possible
- Where fq-* AQMs excel
  - Flow fairness is needed or easy to communicate policies
  - Small number of flows/users
  - CPU available for AQM
- Where using fq-* is challenging
  - High number of flows
  - High speed (e.g. over 40 Gbps)
  - Hierarchical control of Resource Sharing is needed
- ?Buffer size with fq-*

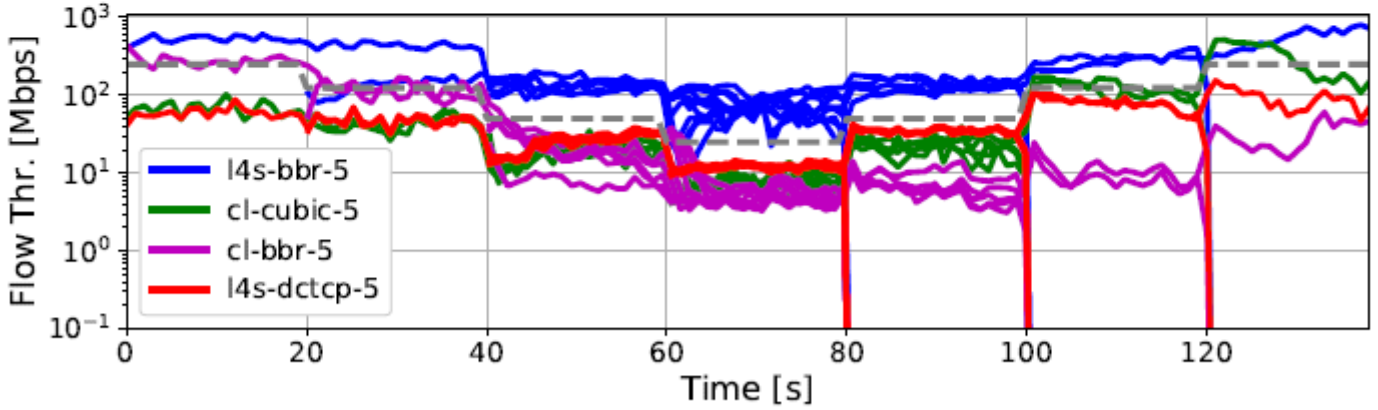
Heterogeneous CCs and equal RTT (5ms)

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

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



VDQ-CSAQM

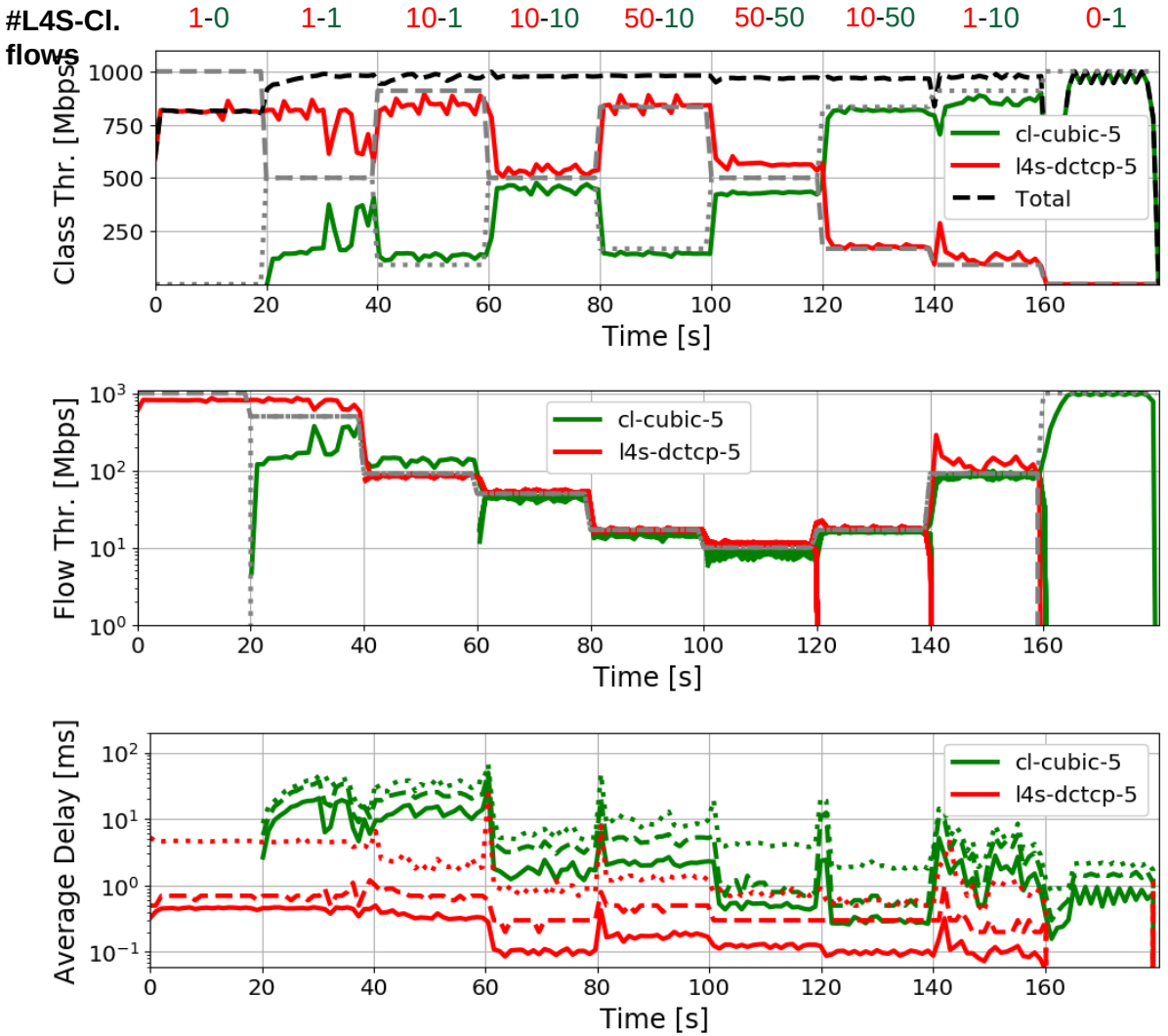


DualPI2

Dynamic traffic – equal RTT (5ms)

DCTCP – Cubic CCs

VDQ-CSAQM



DualPI2