DualQ Coupled AQM

draft: http://www.bobbriscoe.net/projects/latency/draft-briscoe-aqm-dualq-coupled-00.txt
paper: http://www.bobbriscoe.net/projects/latency/dctth_preprint.pdf

Koen De Schepper, Inton Tsang — Bell Labs

Olga Bondarenko — [simula . research laboratory]

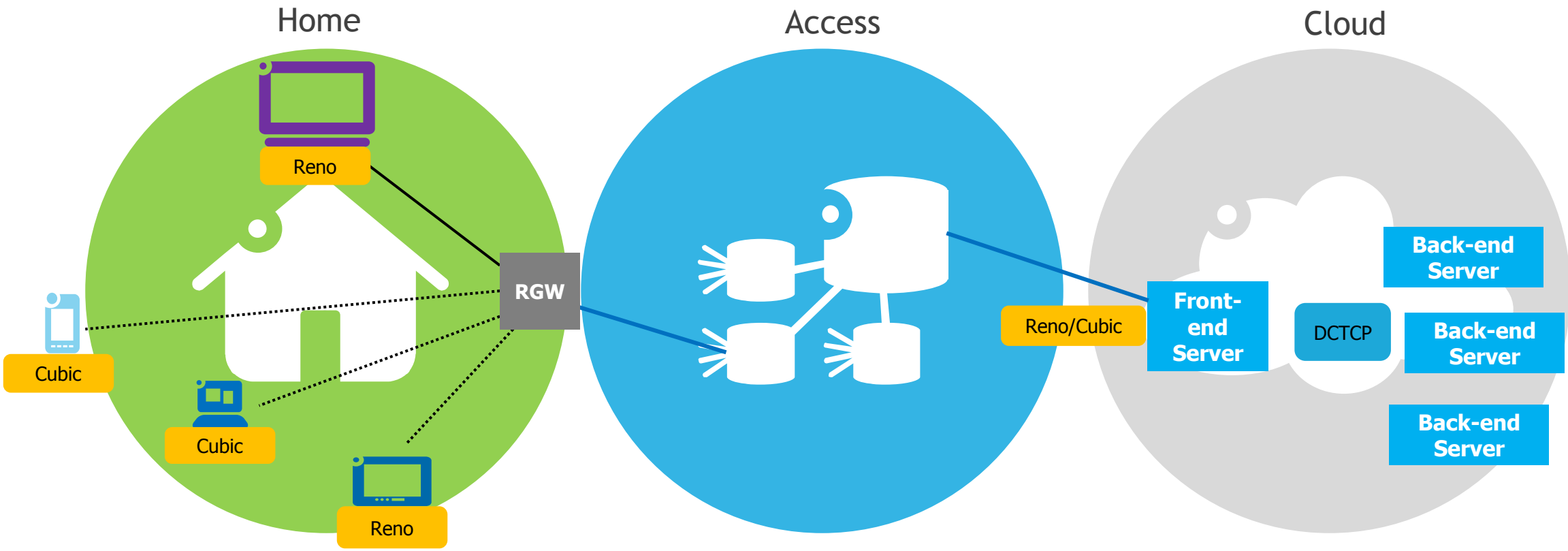
Bob Briscoe — BT

koen.de_schepper@alcatel-lucent.com

July, 2015

R/TE
Reducing Internet Transport Latency

DCTCP = Low Loss, Low Latency, and Throughput Scalability (L4S)

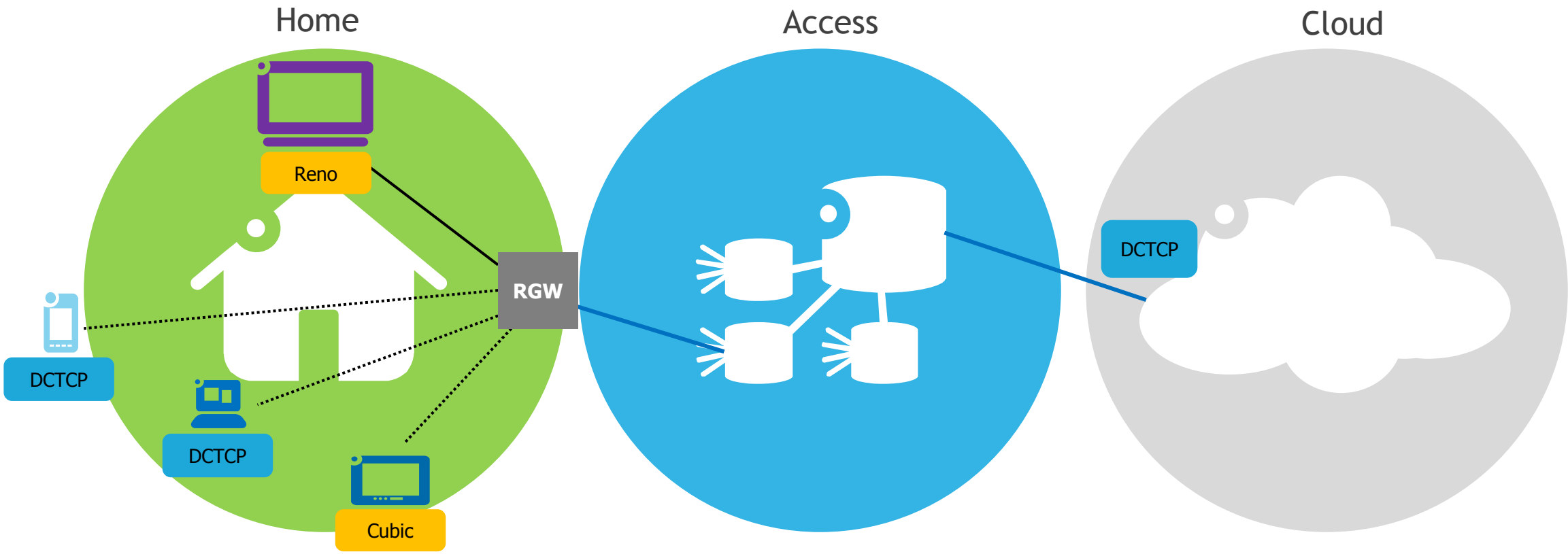


Large queues for high throughput and low drop
= Poor Latency
= Bad for interactive applications

ECN = No drop
ECN++ = Small queues
& Low latency & High throughput

R/TE
Reducing Internet Transport Latency

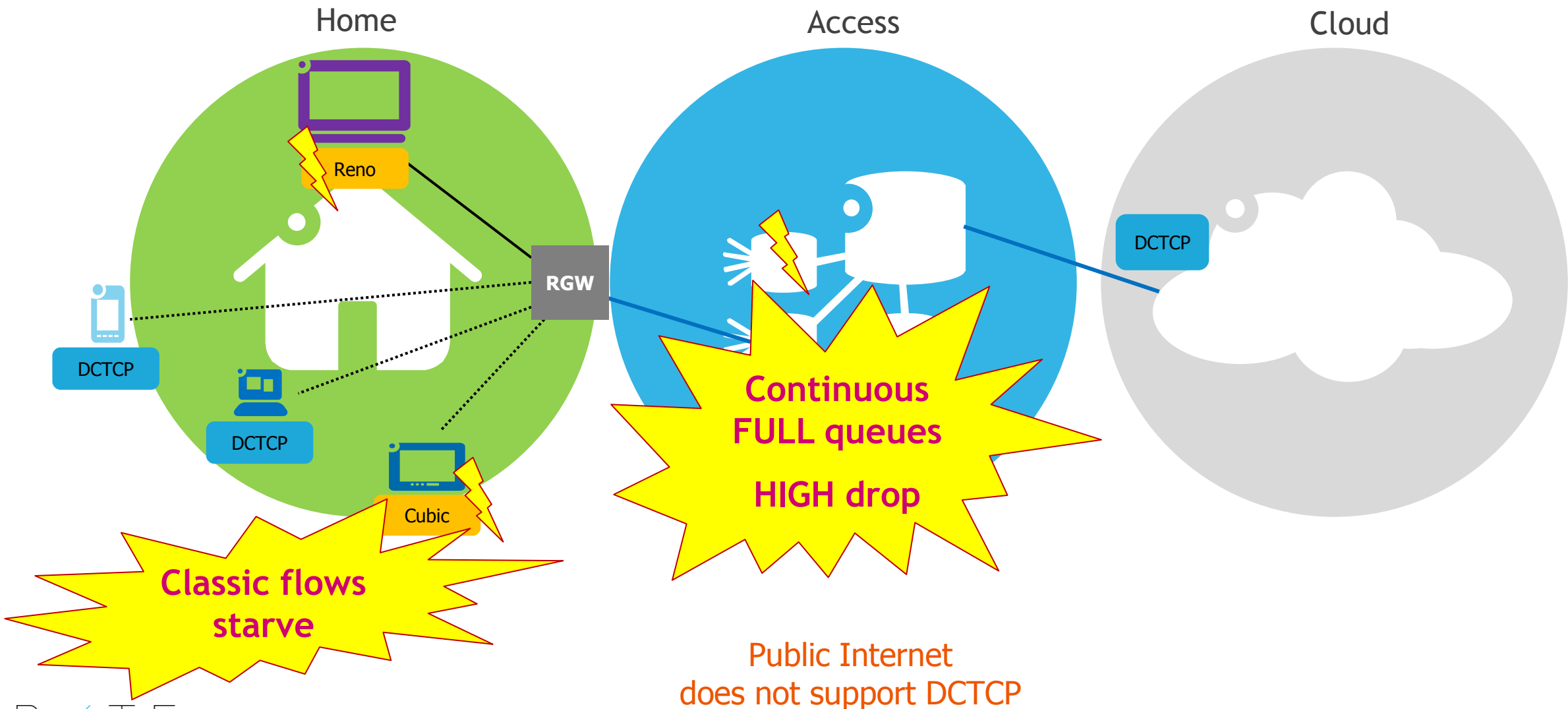
DCTCP to the HOME ?



Windows and Linux 3.18
have DCTCP implementations ready

DCTCP available on
Windows Server and Linux 3.18
used internally in the data center

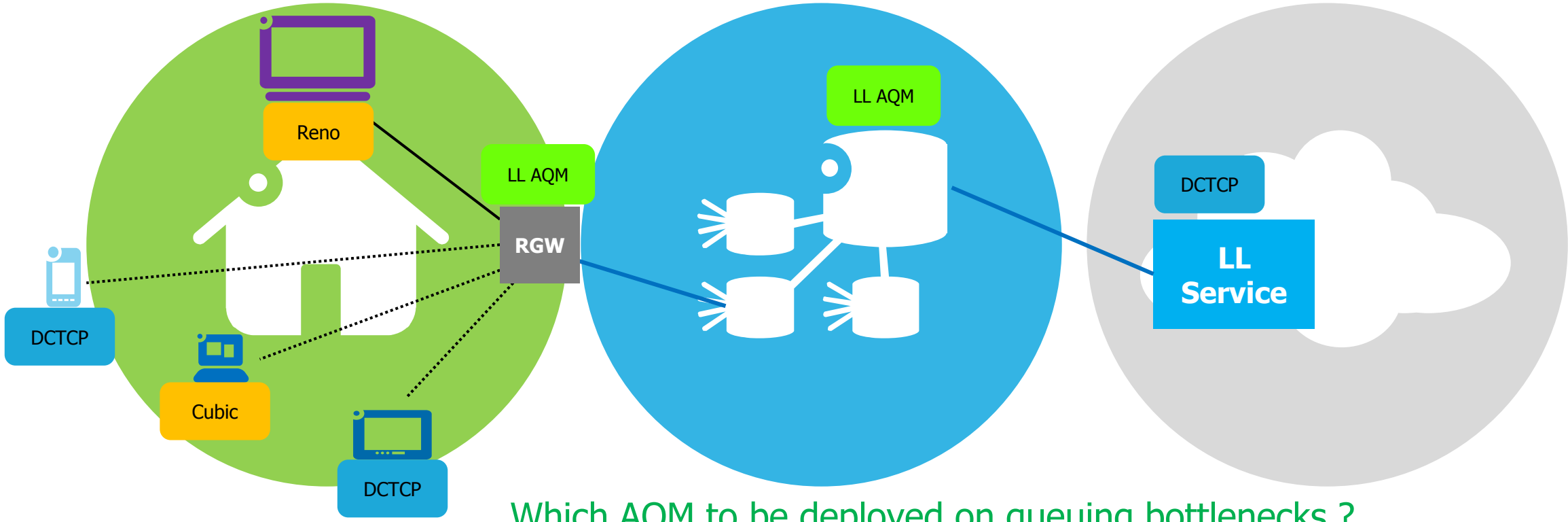
Clients can’t use DCTCP without causing trouble!



Public Internet
does not support DCTCP

R/TE
Reducing Internet Transport Latency

Compatibility Objective: Equal steady state throughput
L4S Objective: Low Latency access to the Cloud

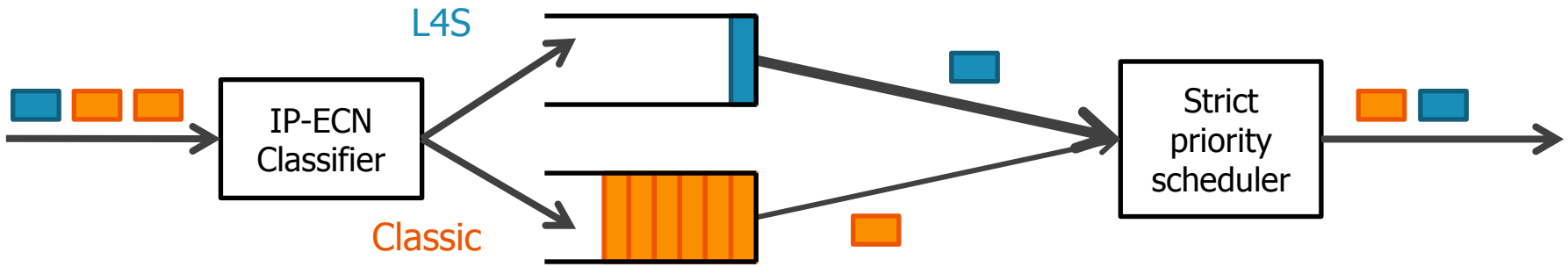


Which AQM to be deployed on queuing bottlenecks ?

Support migration to L4S for all!

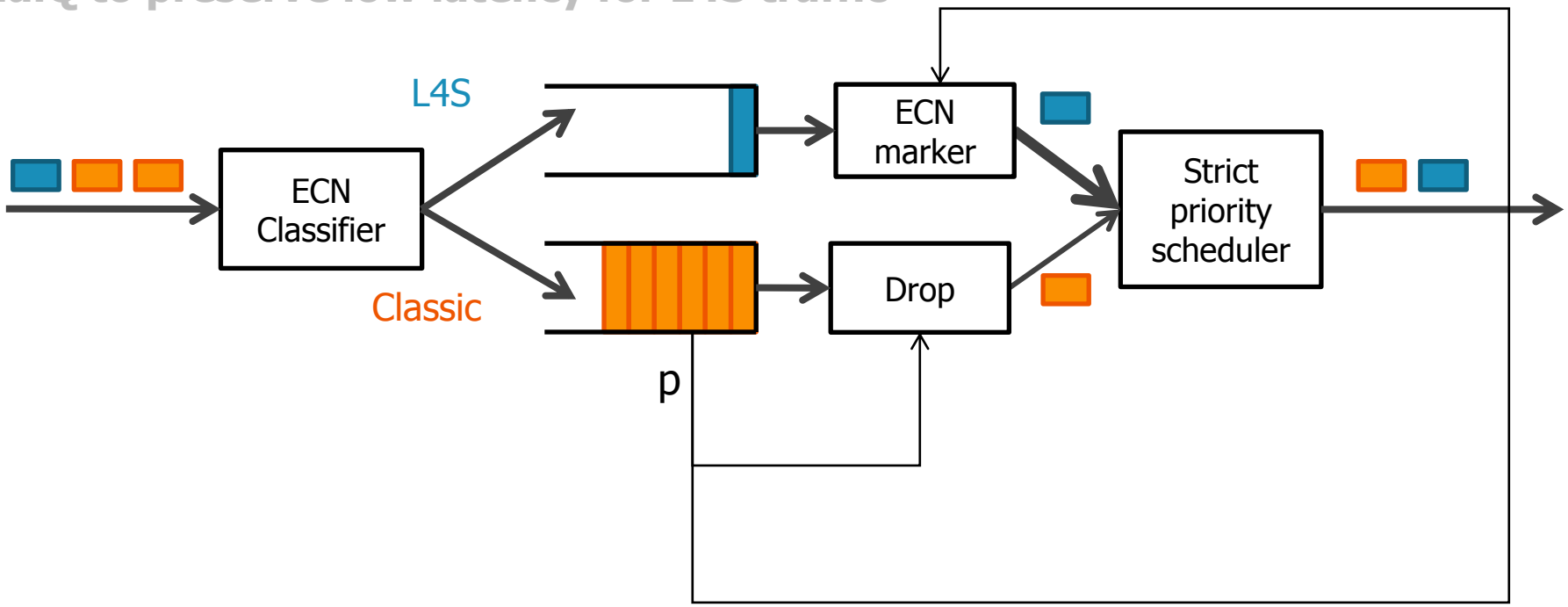
Dual Queue Coupled AQM
Concept 1: DualQ

DualQ to preserve low latency for L4S traffic



Dual Queue Coupled AQM
Concept 2: Coupled AQM

DualQ to preserve low latency for L4S traffic



Coupled AQM to control priority traffic

Dual Queue Coupled AQM
Concept 3: Don't Think Twice to mark

DON'T Think twice to mark

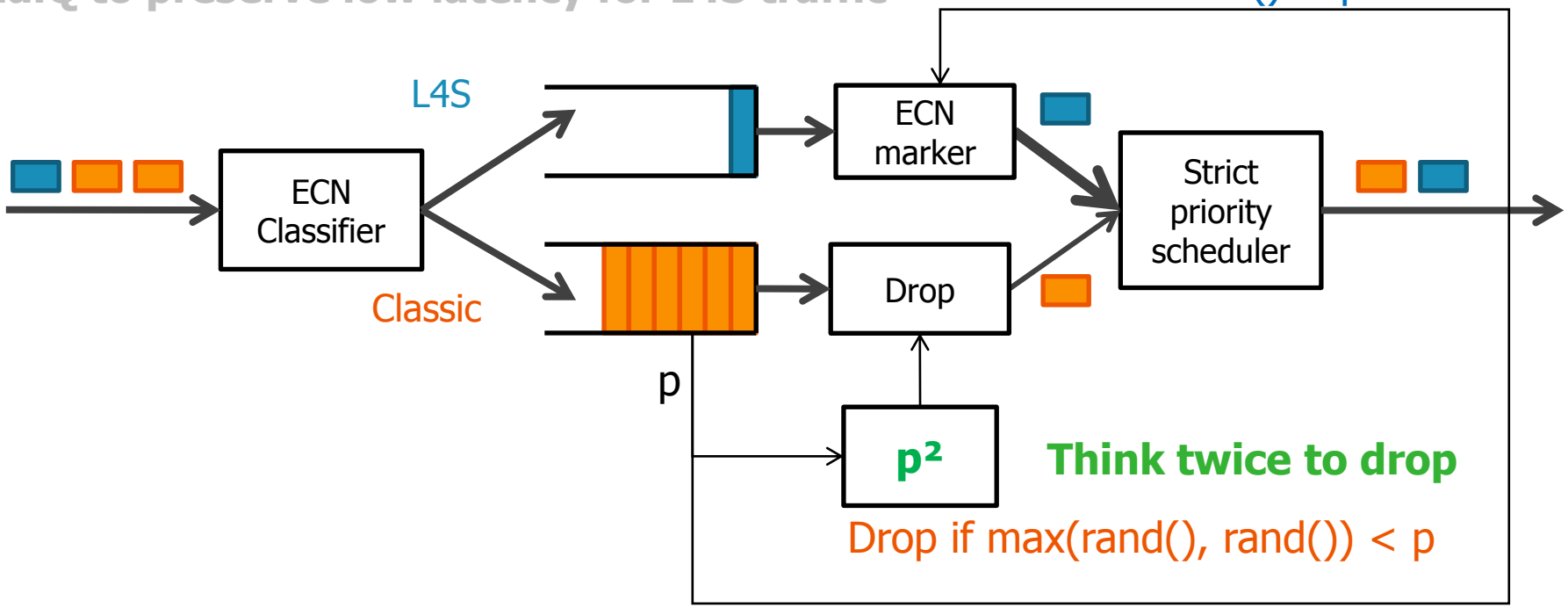
DualQ to preserve low latency for L4S traffic

L4S (DCTCP)

$$r \approx 1/p$$

Classic (Reno / Cubic)

$$r \approx 1/\sqrt{p}$$

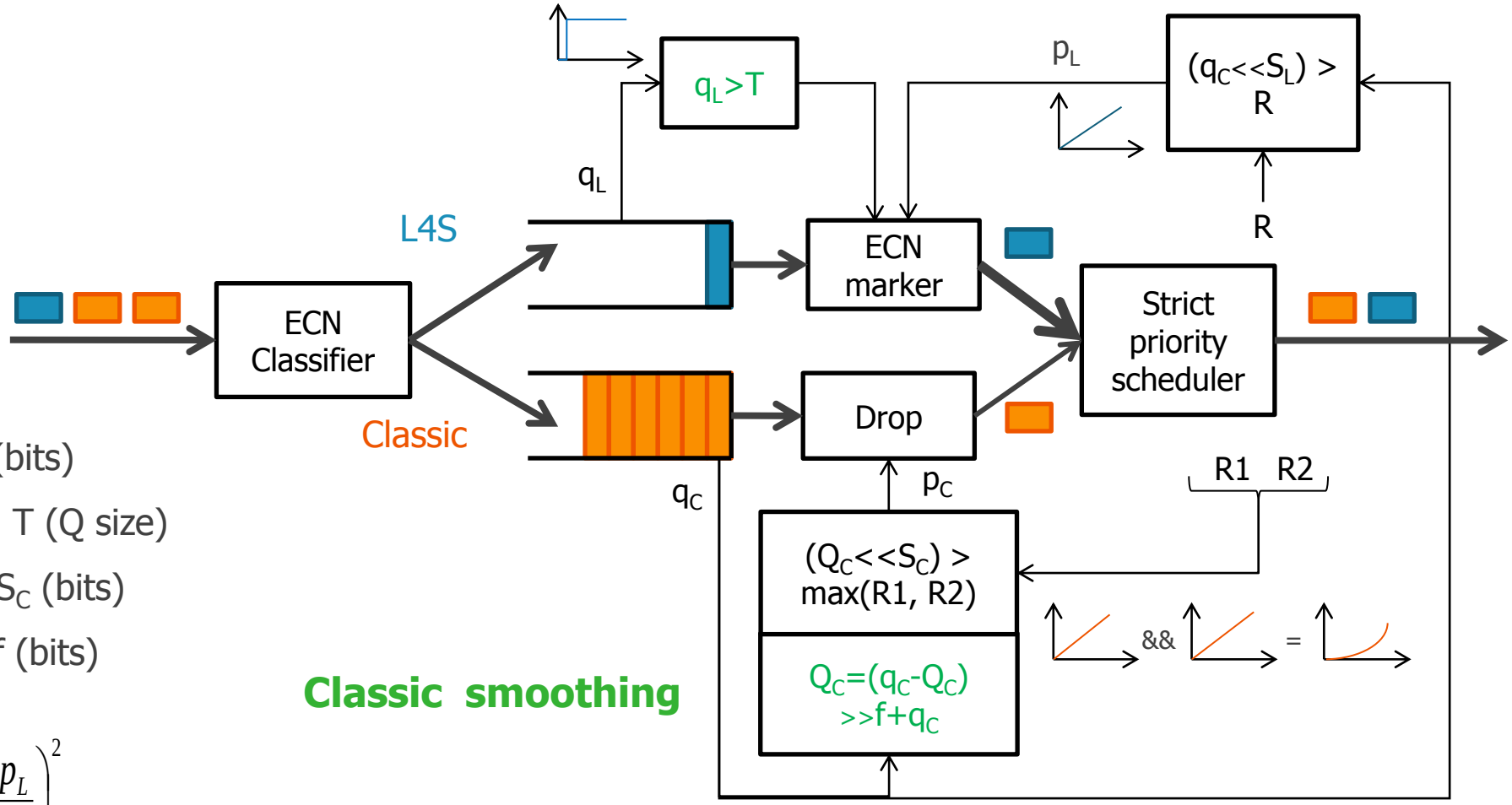


Coupled AQM for equal rate

Detailed Implementation 2 Details

NO SMOOTHING for L4S

L4S AQM if no Classic traffic



4 parameters:

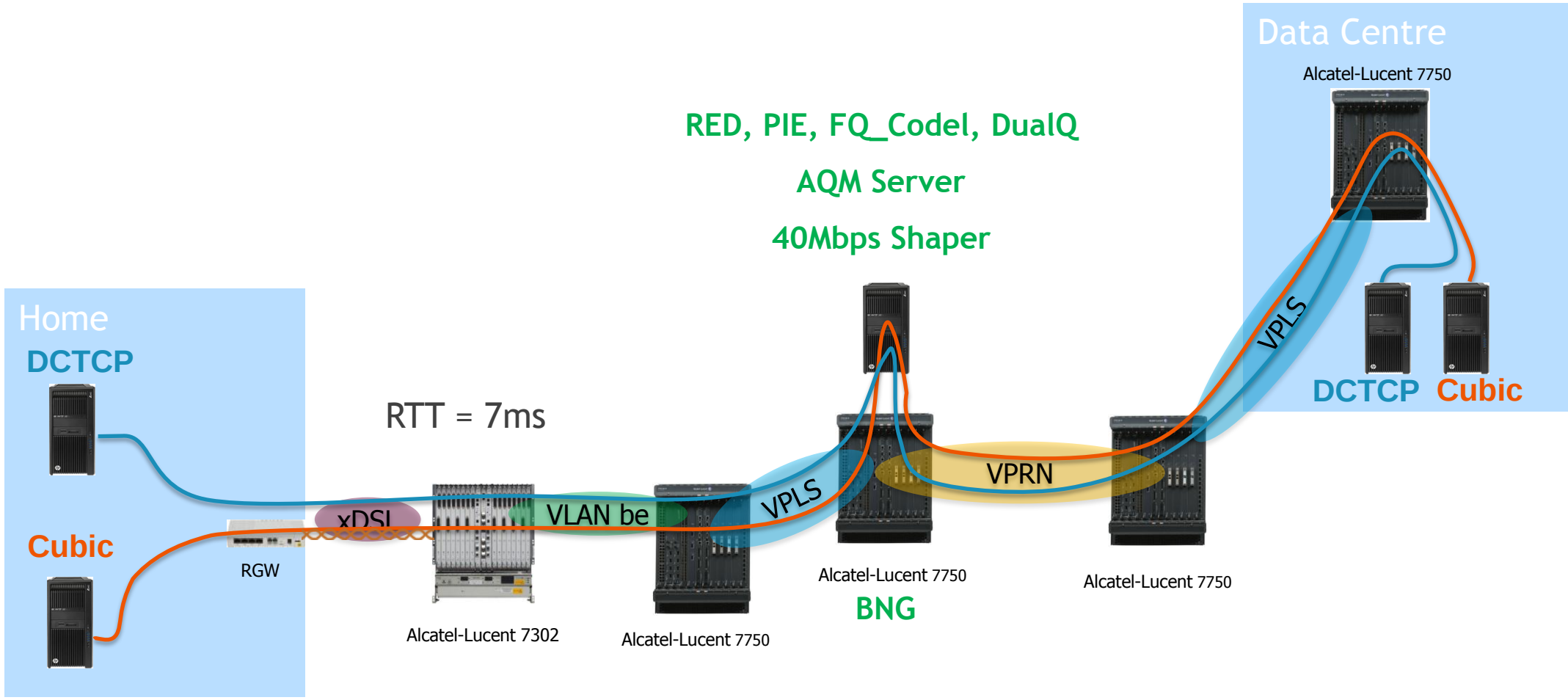
- L4S slope S_L (bits)
- L4S threshold T (Q size)
- Classic slope S_C (bits)
- EWMA value f (bits)

Classic smoothing

$k = S_L - S_C$

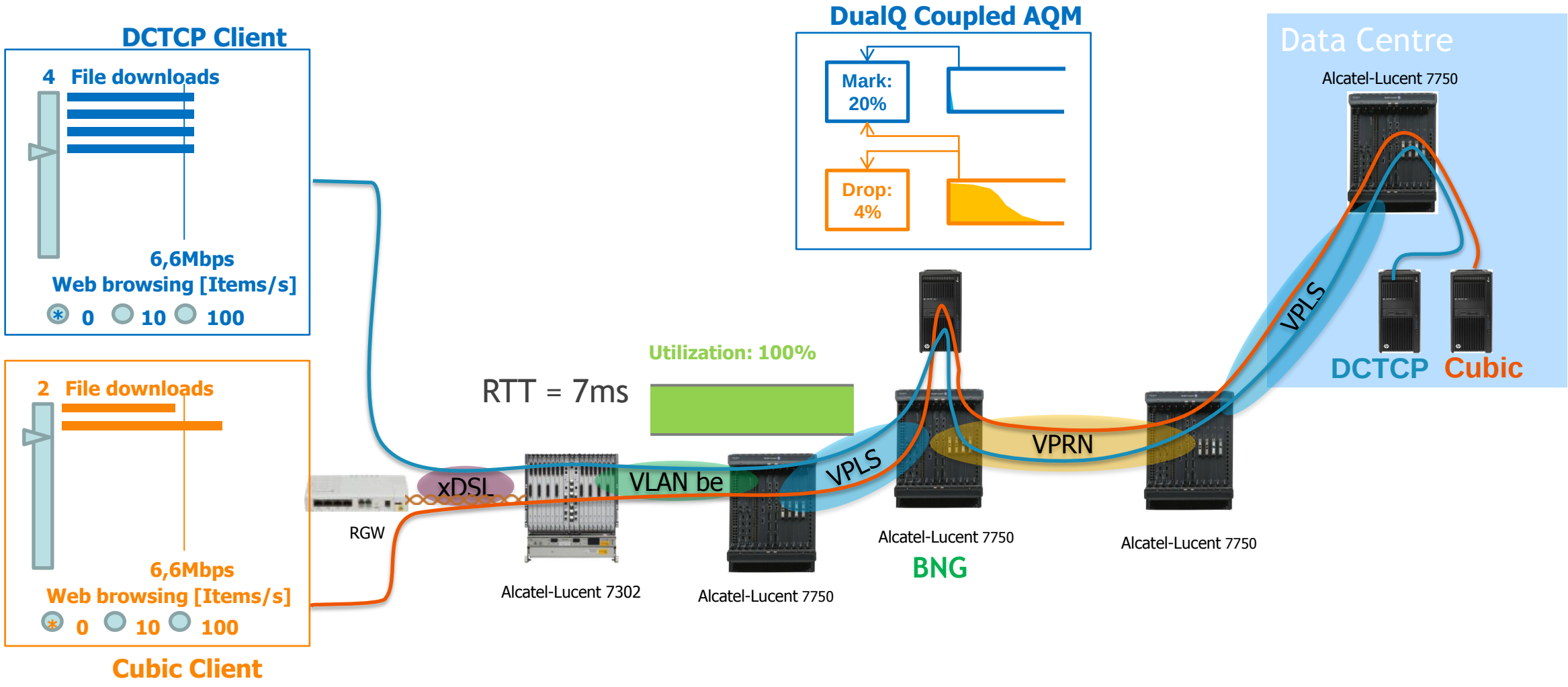
Coupling: $p_C = \left(\frac{p_L}{2^k}\right)^2$

Evaluated on a Real BB Residential Testbed



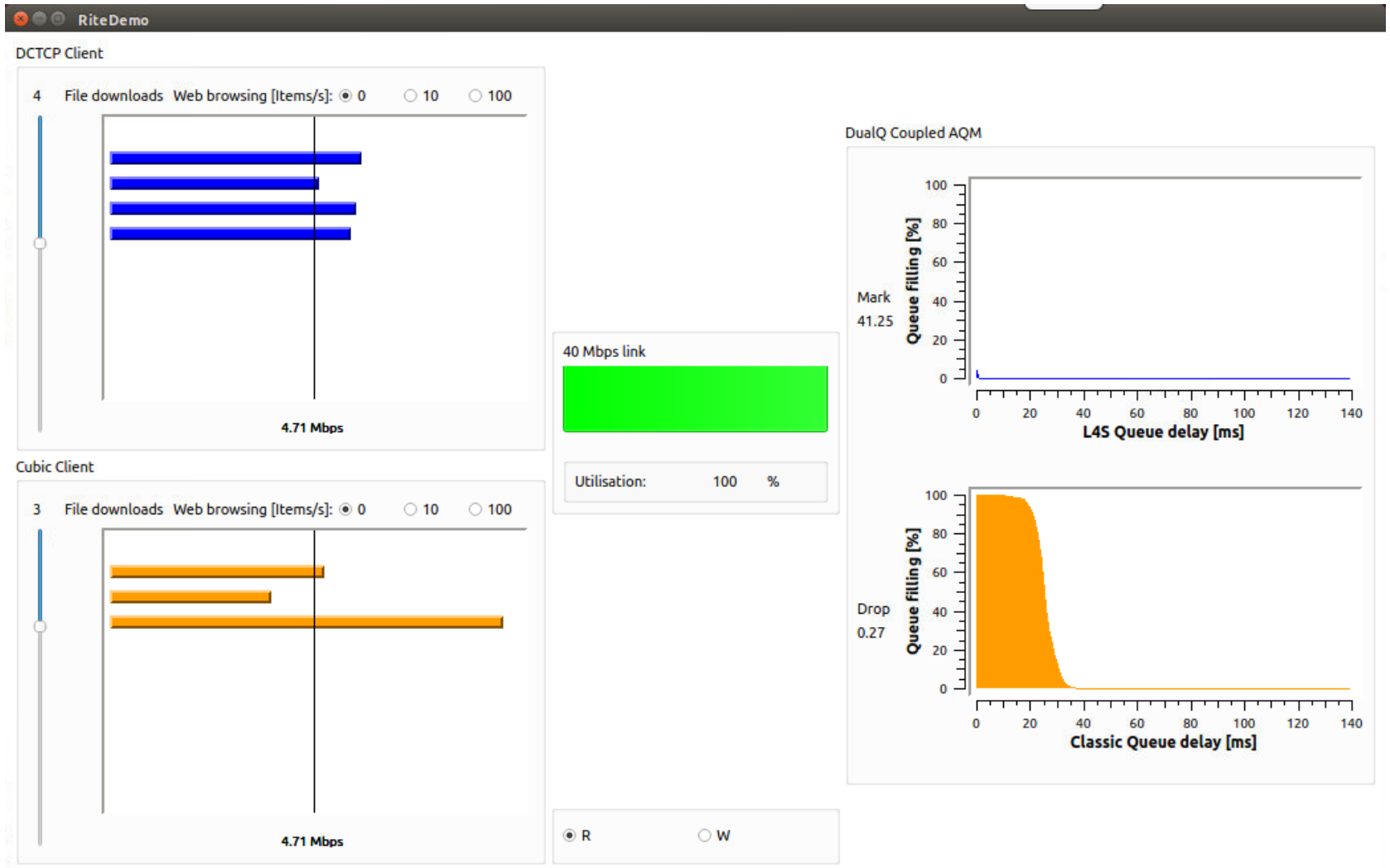
Demo on a Real BB Residential Testbed

All values are measured from a live traffic capture



Dual Queue Coupled AQM Demonstration

All values are measured from a traffic capture



Steady state Equal rate Test Scenarios

- Evaluate throughput equality
- Combination of each 1 to 10 flows, running for 250 seconds

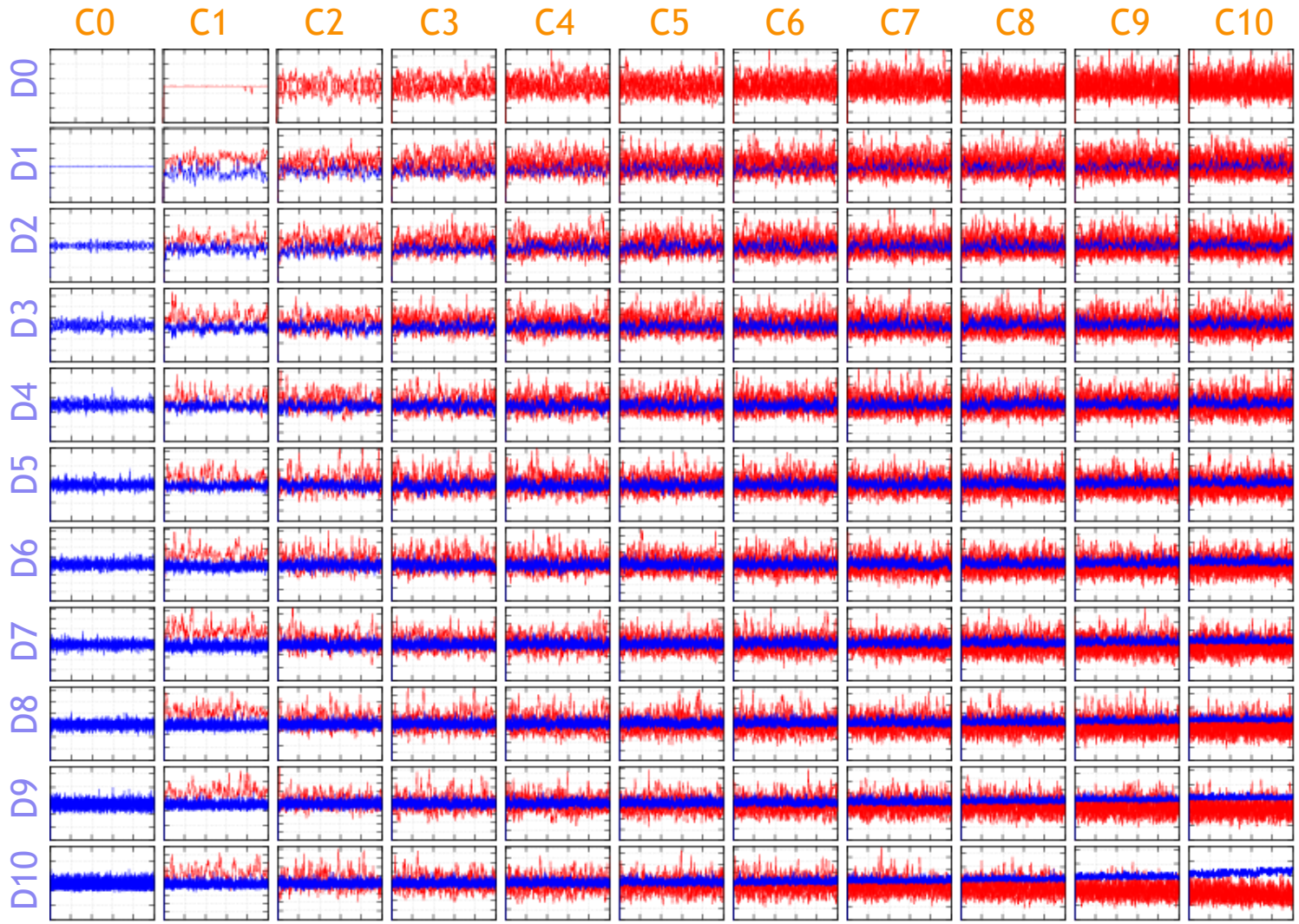
THROUGHPUT DUALQ

D = DCTCP C = CUBIC

X = [0:250] sec

Y = [0:80/N] Mbps

N = nbr of flows



→Equal throughput between DCTCP and Cubic/Reno

→Only 2 queues

R/TE
Reducing Internet Transport Latency

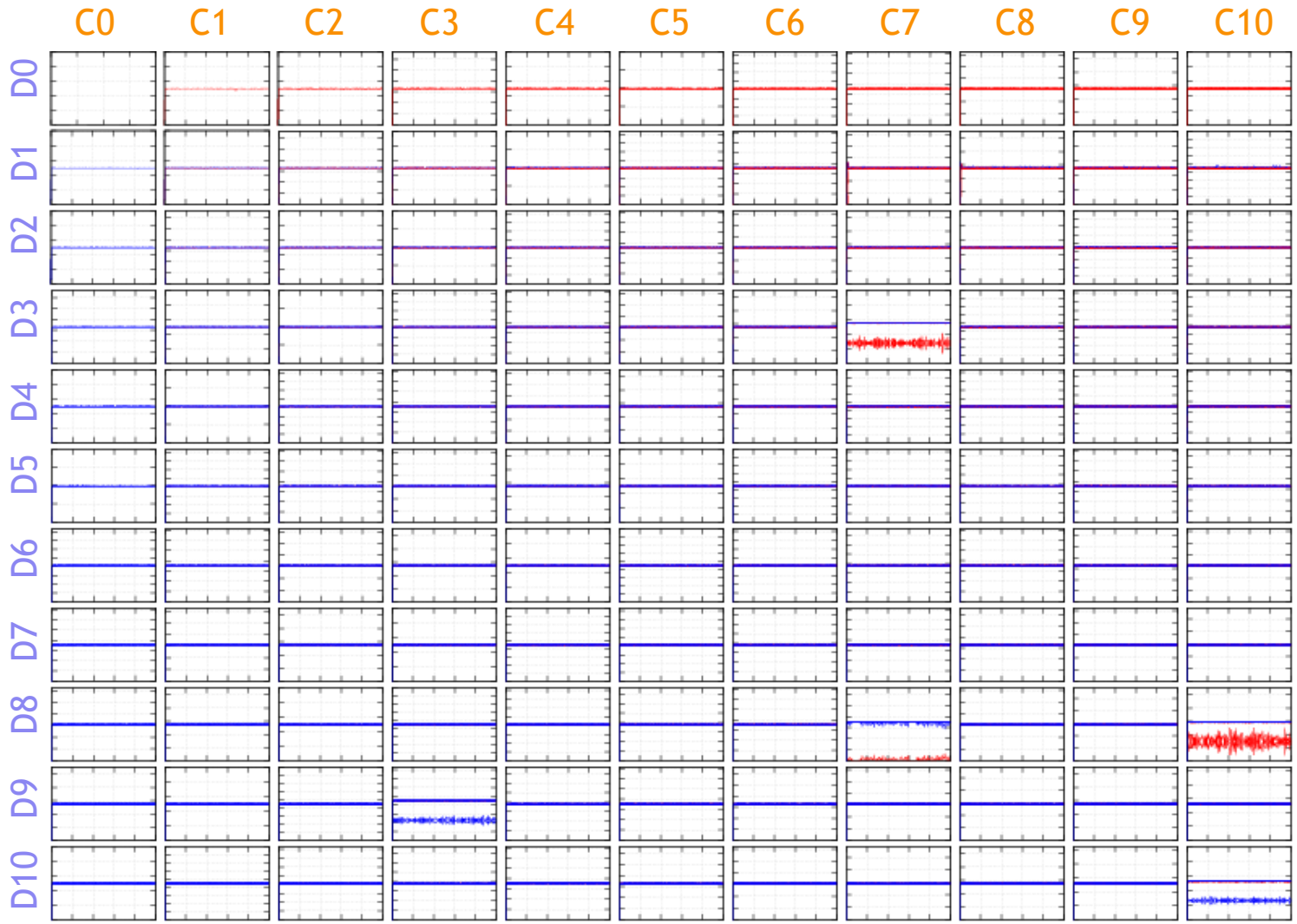
THROUGHPUT FQ_CODEL*

D = DCTCP C = CUBIC

X = [0:250] sec

Y = [0:80/N] Mbps

N = nbr of flows



→Equal throughput as Qs are large enough

→Accidental Q conflicts (D8/C7) (6,5 expected)

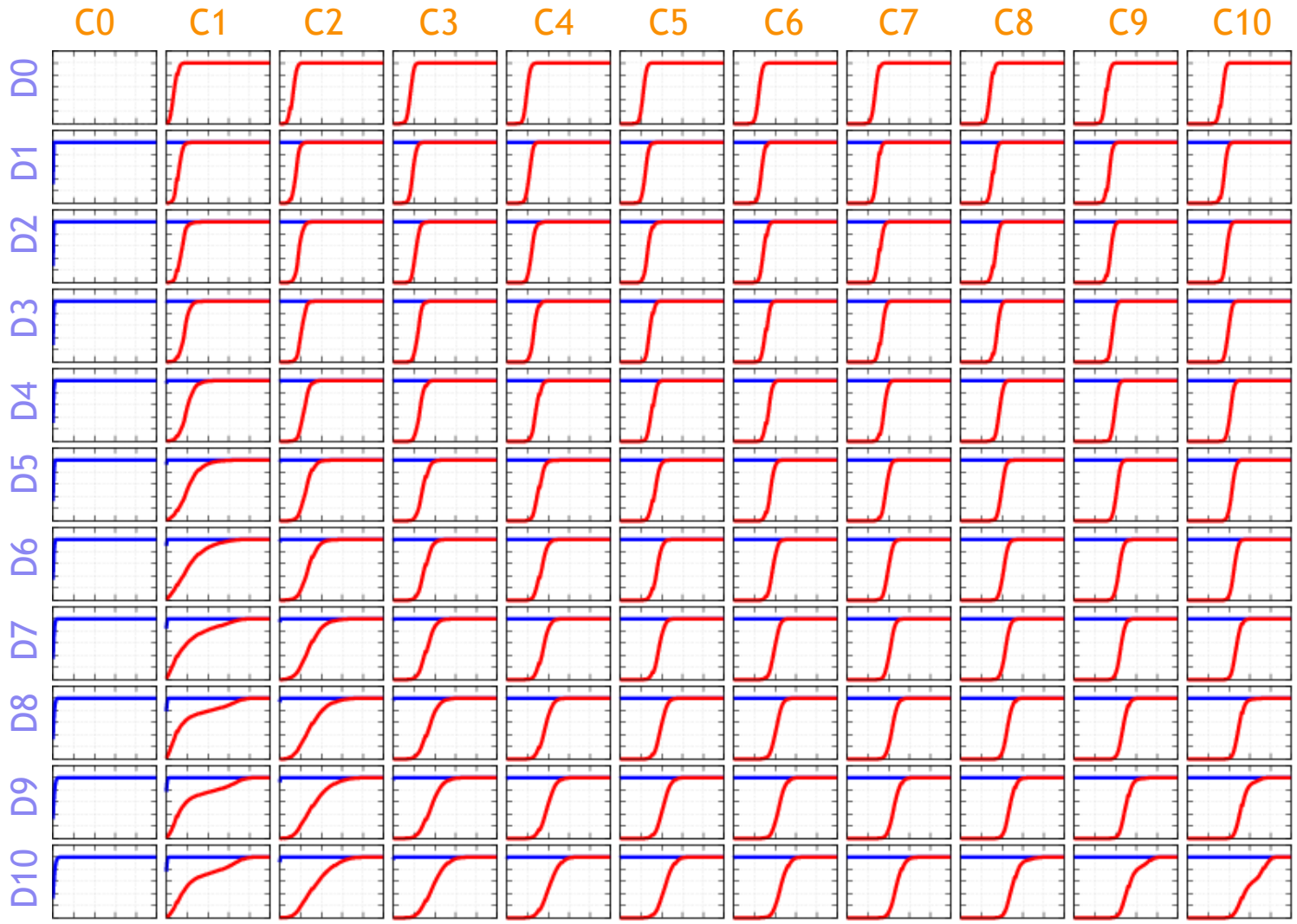
* tc qdisc add dev eth2 root fq_codel

QSIZE CDF DUALQ

D = DCTCP C = CUBIC

X = [0:100] ms

Y = [0:120] %



→Extreme small queue size for DCTCP

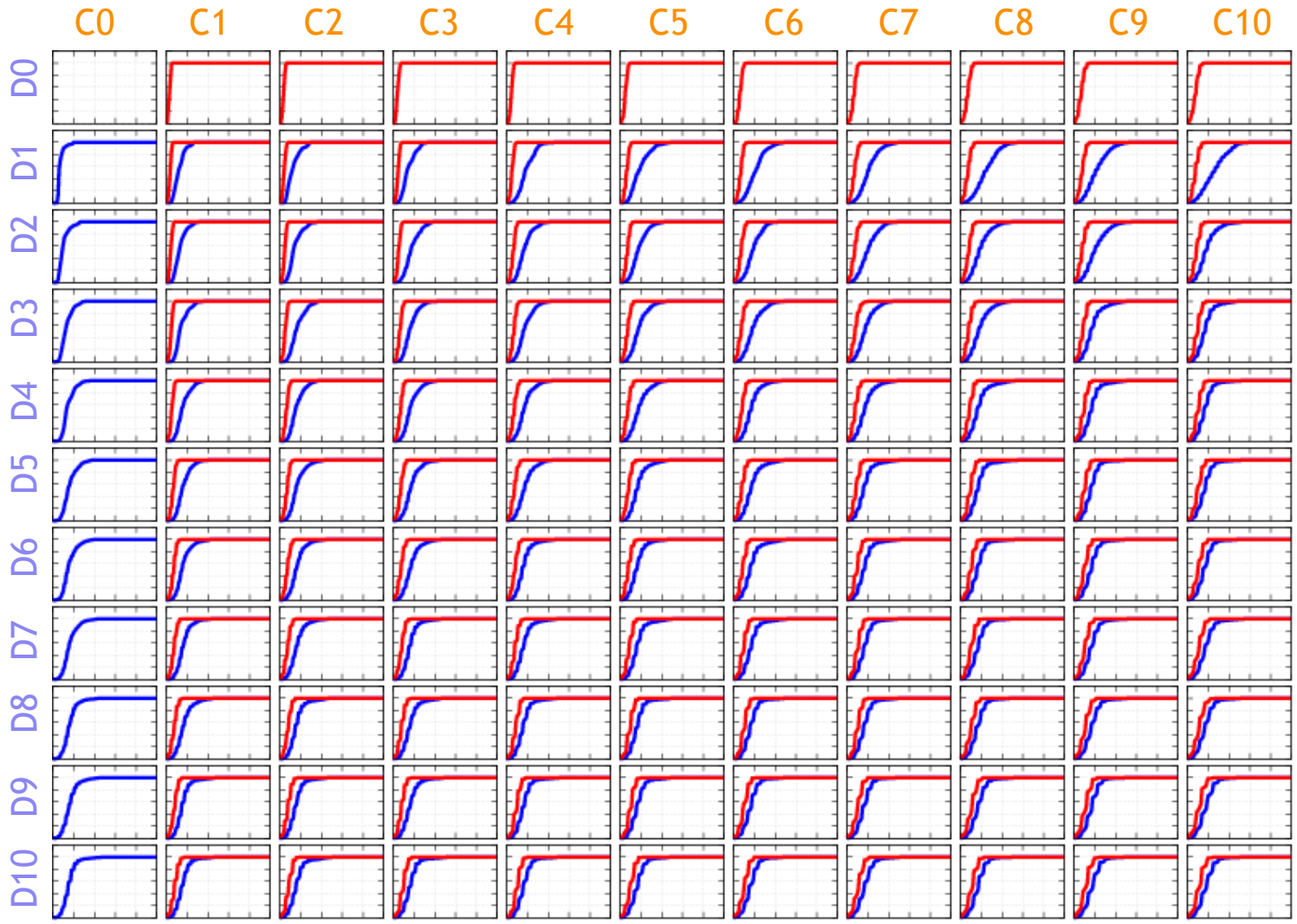
→Classic queue size defined by classic AQM

QSIZE CDF FQ_CODEL*

D = DCTCP C = CUBIC

X = [0:100] ms

Y = [0:120] %



→ FQ_Codel can not achieve its stricter target Q size

* tc qdisc add dev eth2 root fq_codel

Dynamic Low Latency Test Scenarios

- Evaluate End User Latency for different Flow sizes and Load levels

- Exponential arrival distribution
  - Low load (L = 10 requests per second)
  - High load (H = 100 requests per second)

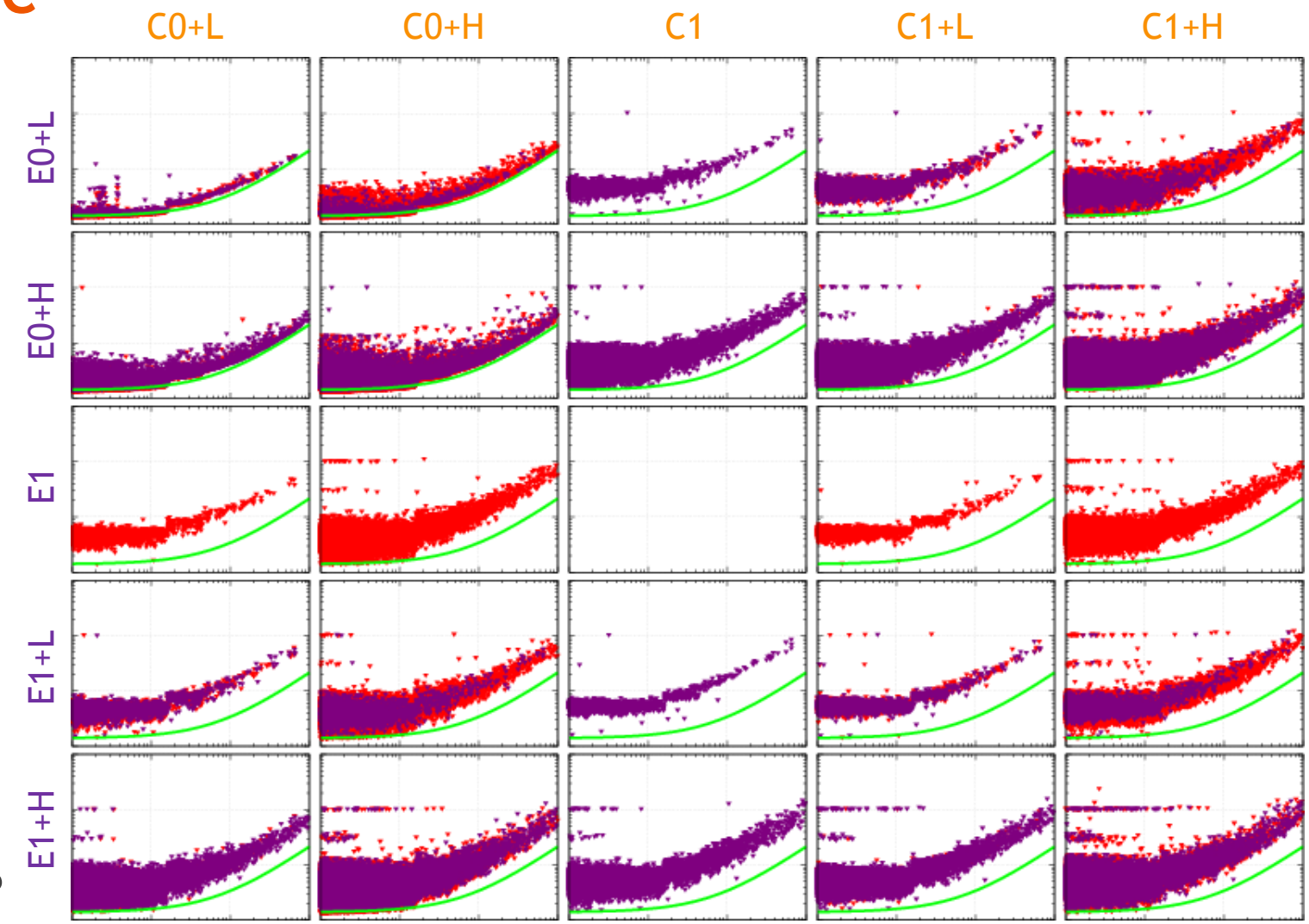
- Size distribution
  - Pareto 1KB to 1MB

COMPLETION TIMES PIE

E = CUBIC+ECN C = CUBIC

X = log [0.001:1] MB

Y = log [0.01:10] s



→ECN improves by preventing longest delays caused by retransmission timeouts

→Burst allowance not effective when long flows are active

→No ECN on SYN → 1s timeout

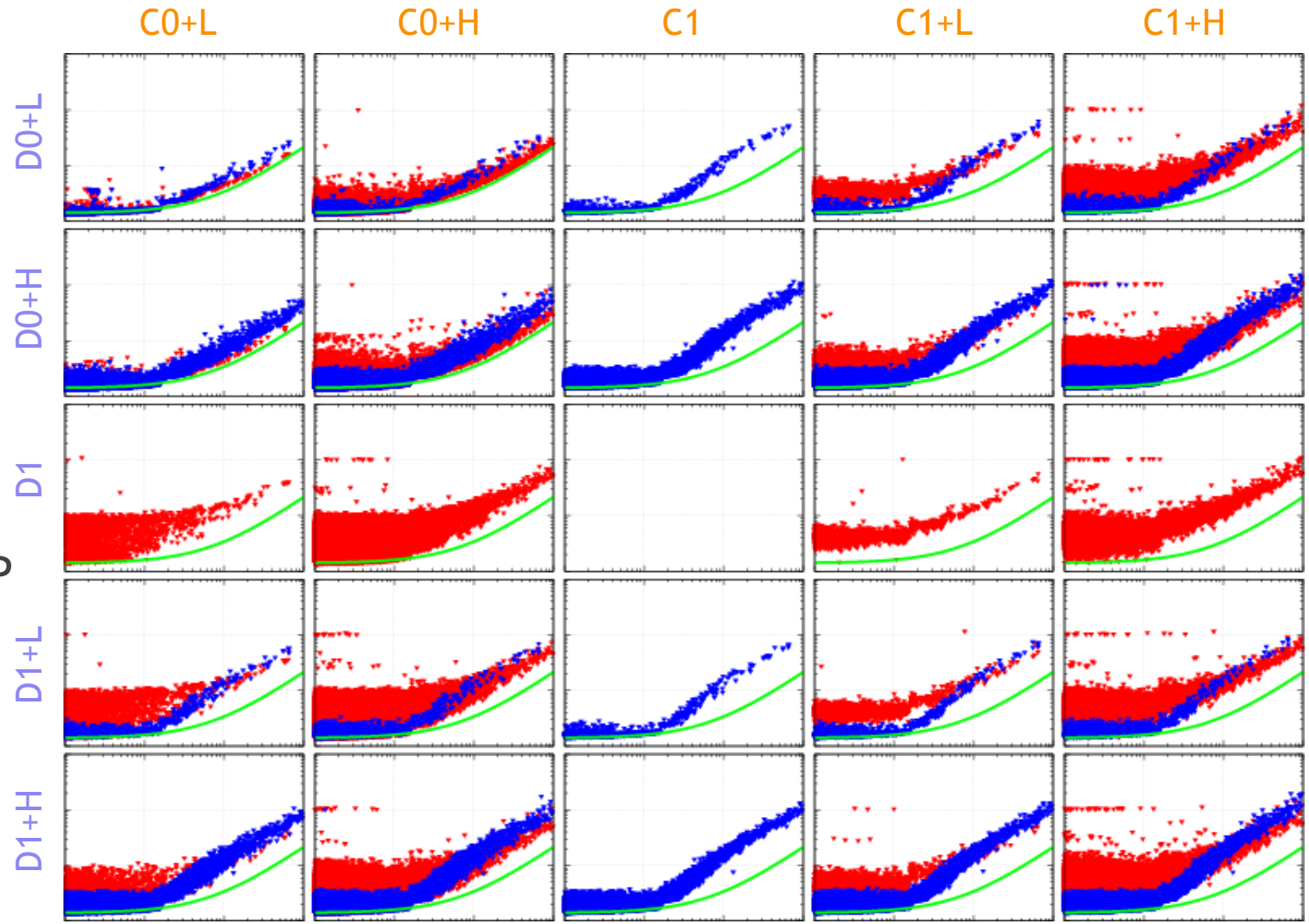
→ECN issue in Linux for sizes < 3KB

COMPLETION TIMES DUALQ

D = DCTCP C = CUBIC

X = log [0.001:1] MB

Y = log [0.01:10] s



→ Consistent low latency for DCTCP

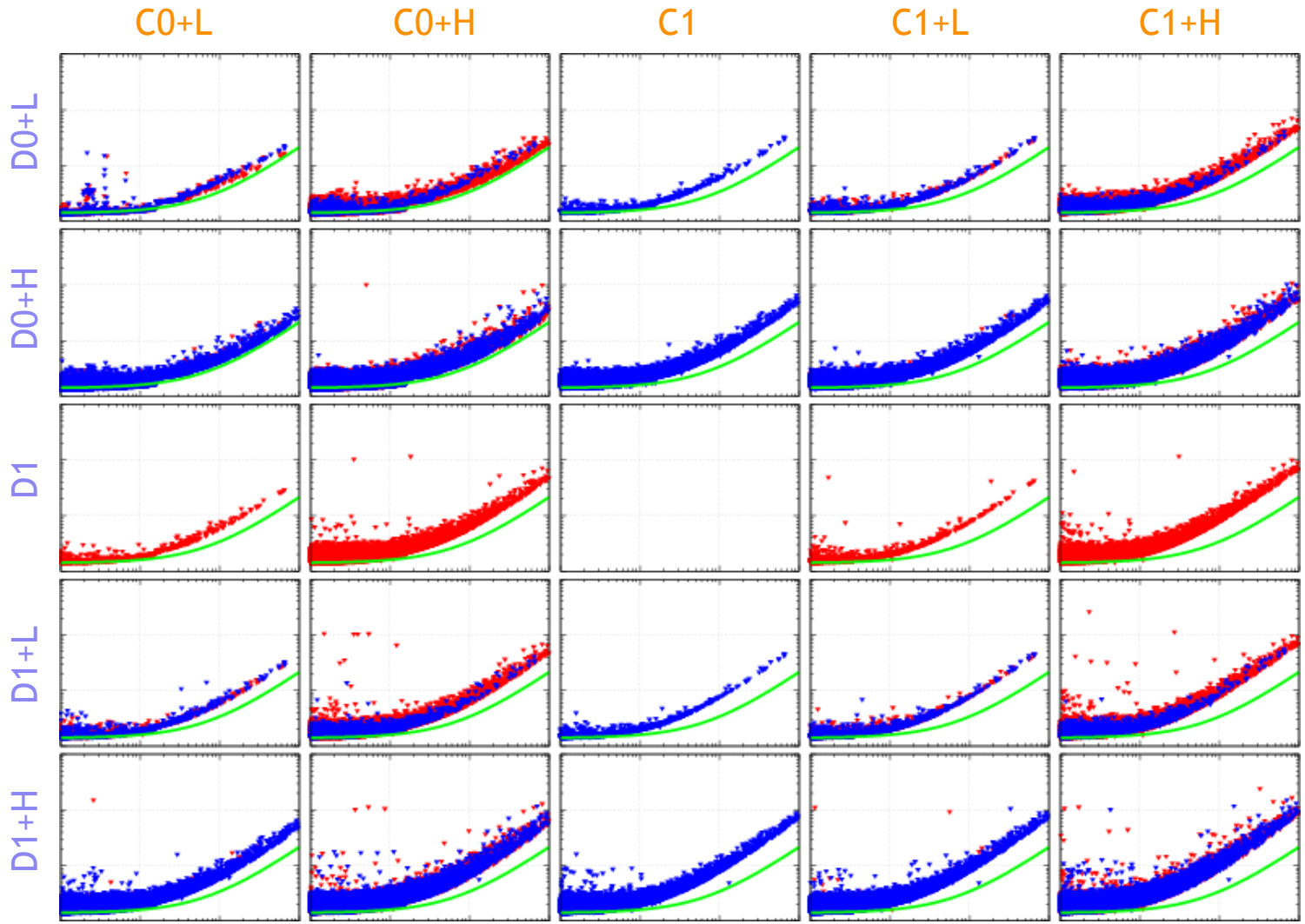
→ DCTCP flows get less throughput due to exiting too fast slow start and smaller classic queue

COMPLETION TIMES FQ_CODEL

D = DCTCP C = CUBIC

X = log [0.001:1] MB

Y = log [0.01:10] s



→ Almost perfect scheduled

→ Burst allowance is effective due to separate queues, except when collisions

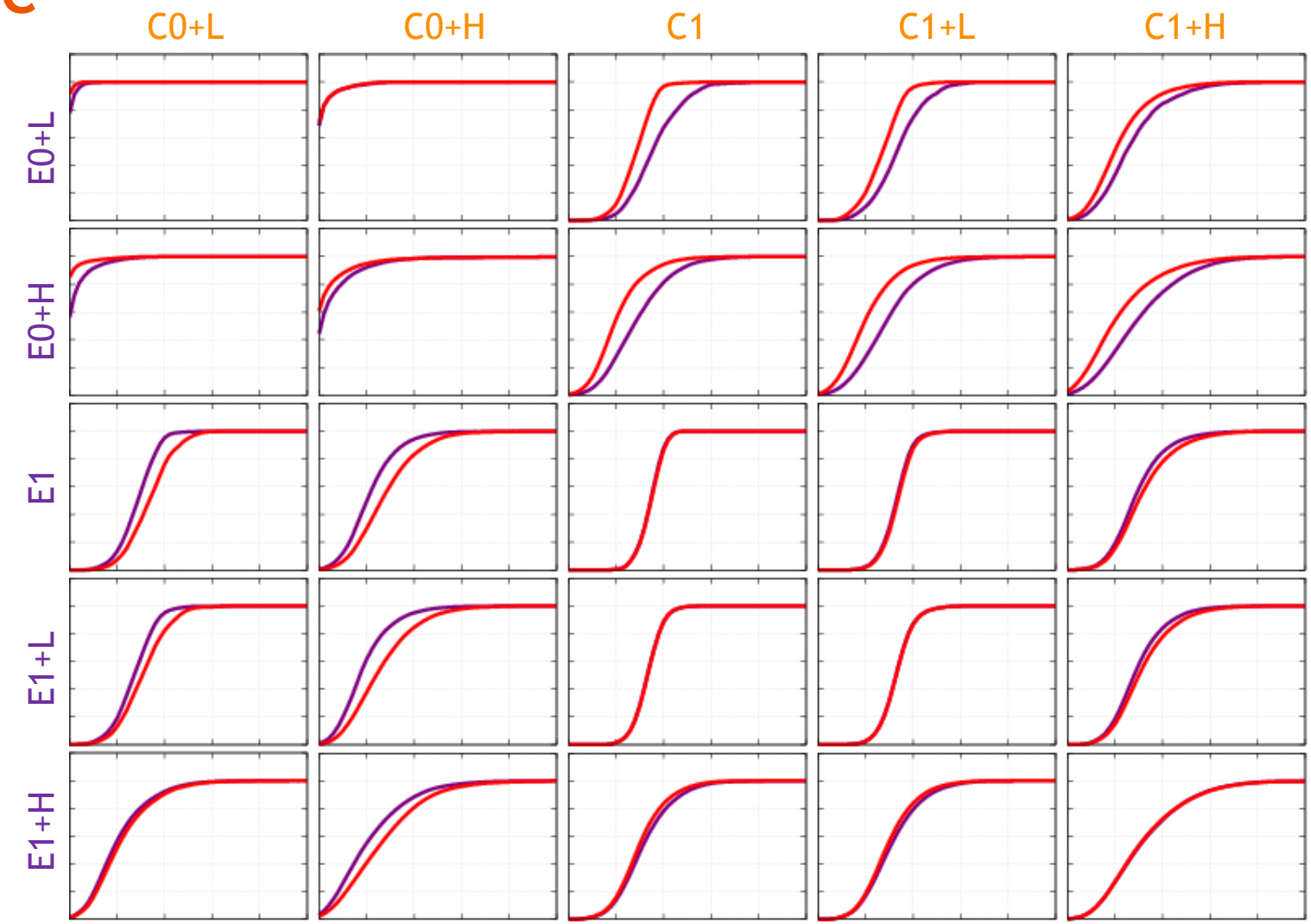
QSIZE CDF

PIE

E = CUBIC+ECN C = CUBIC

X = [0:50] ms

Y = [0:120] %



→ Q sizes around 20ms when long flows are active

→ ECN packets experience slightly longer delay in queue because not dropped

QSIZE CDF

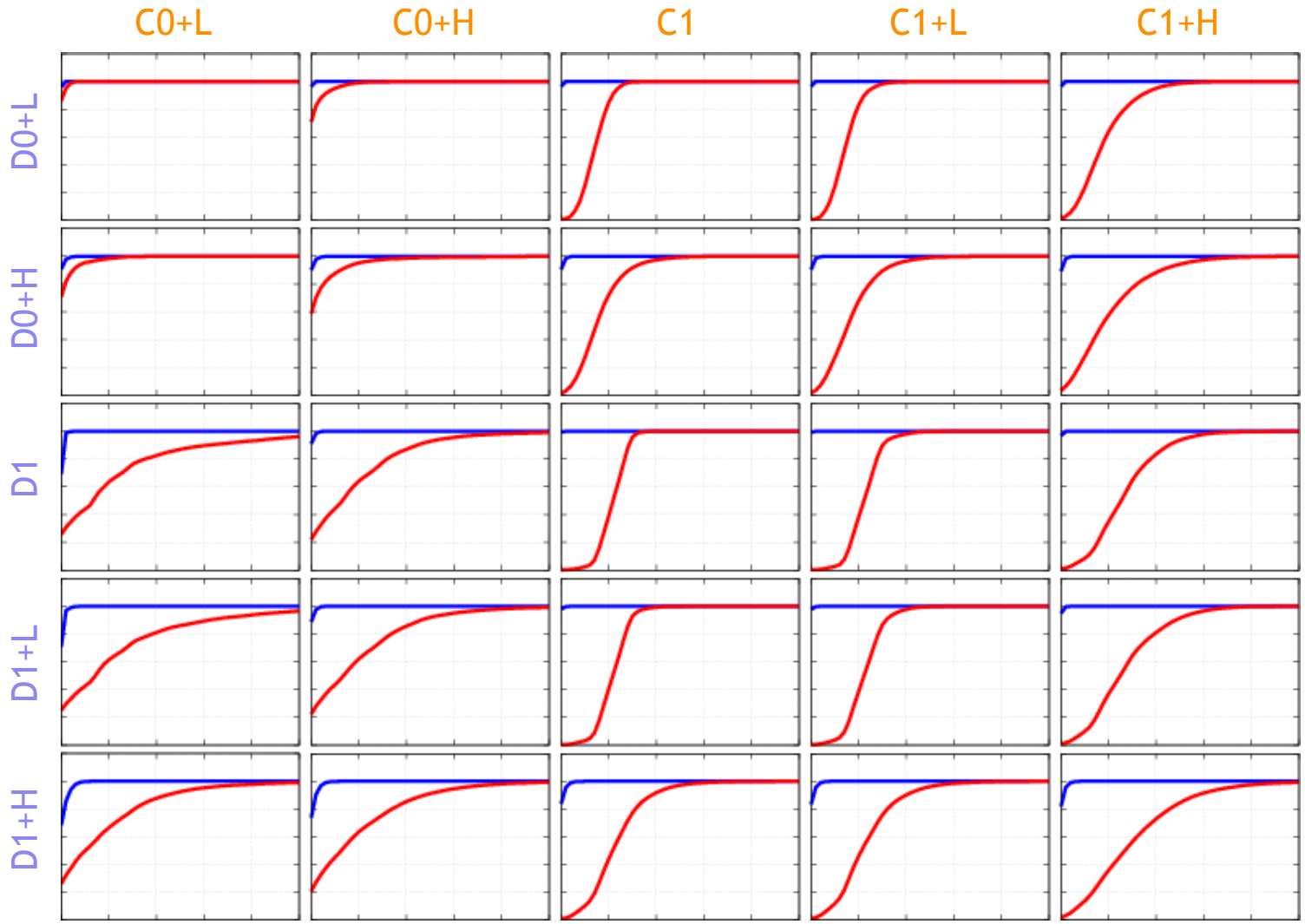
DUALQ

D = DCTCP C = CUBIC

X = [0:50] ms

Y = [0:120] %

→ Extreme small queues for DCTCP
no slow start overshoot



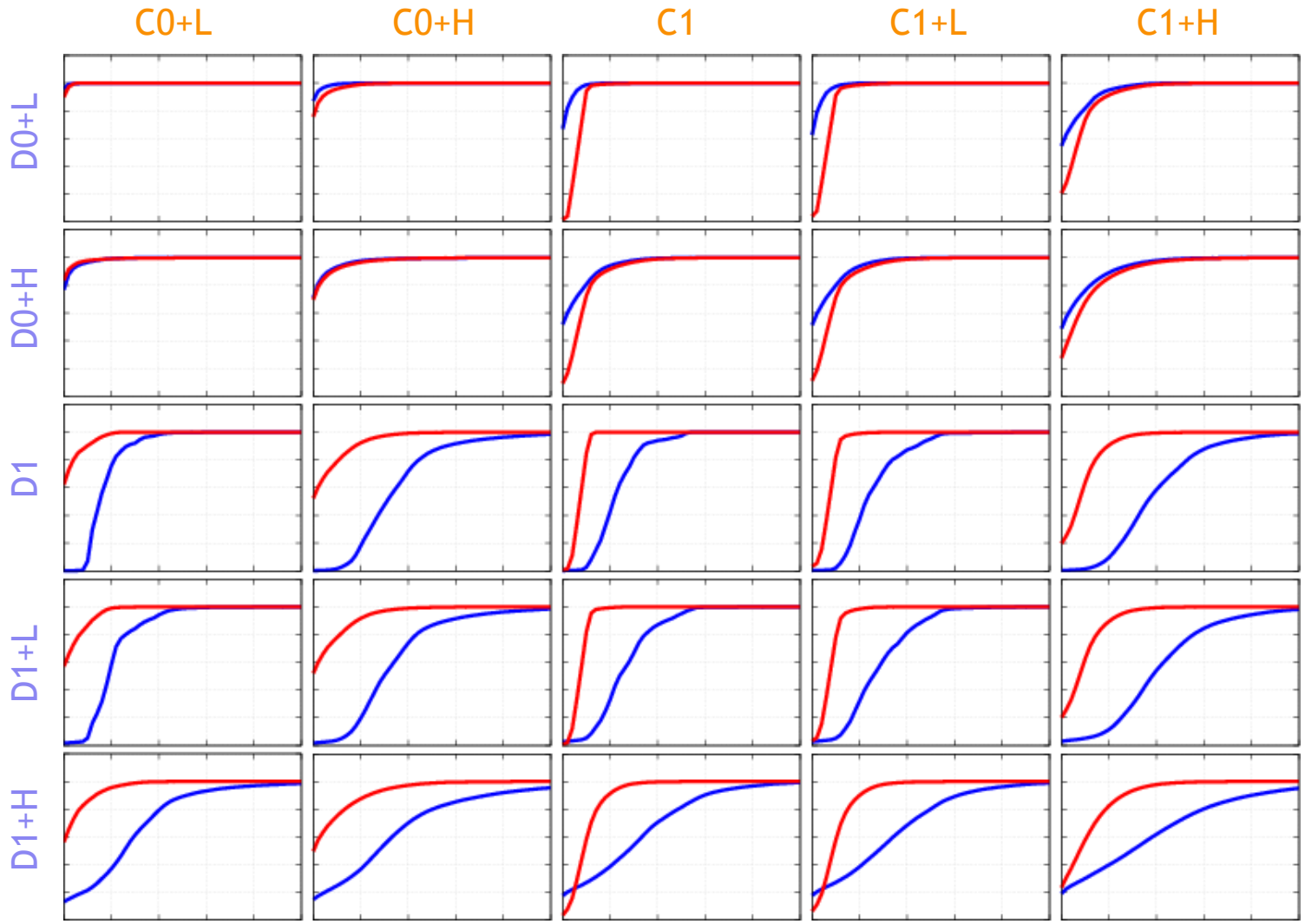
QSIZE CDF

FQ_CODEL

D = DCTCP C = CUBIC

X = [0:50] ms

Y = [0:120] %



→ DCTCP has larger self inflicted queue

DCttH Objective: Towards Universal Support for Low Latency

Make DCTCP safer to use

- Less safe DCTCP could be considered for:
  - Coexistence in Data Centres
  - Controlled / Walled garden deployments like DCttH

- For Public Internet use:
  - Fall back to Reno or Cubic behaviour on loss
  - Handle a window of less than 2 when the RTT is low
  - ECN negotiation issues (compatibility with Classic ECN)
  - Detection of Classic ECN routers in the network (if large delays are experienced)
  - DCTCP support for larger RTTs?

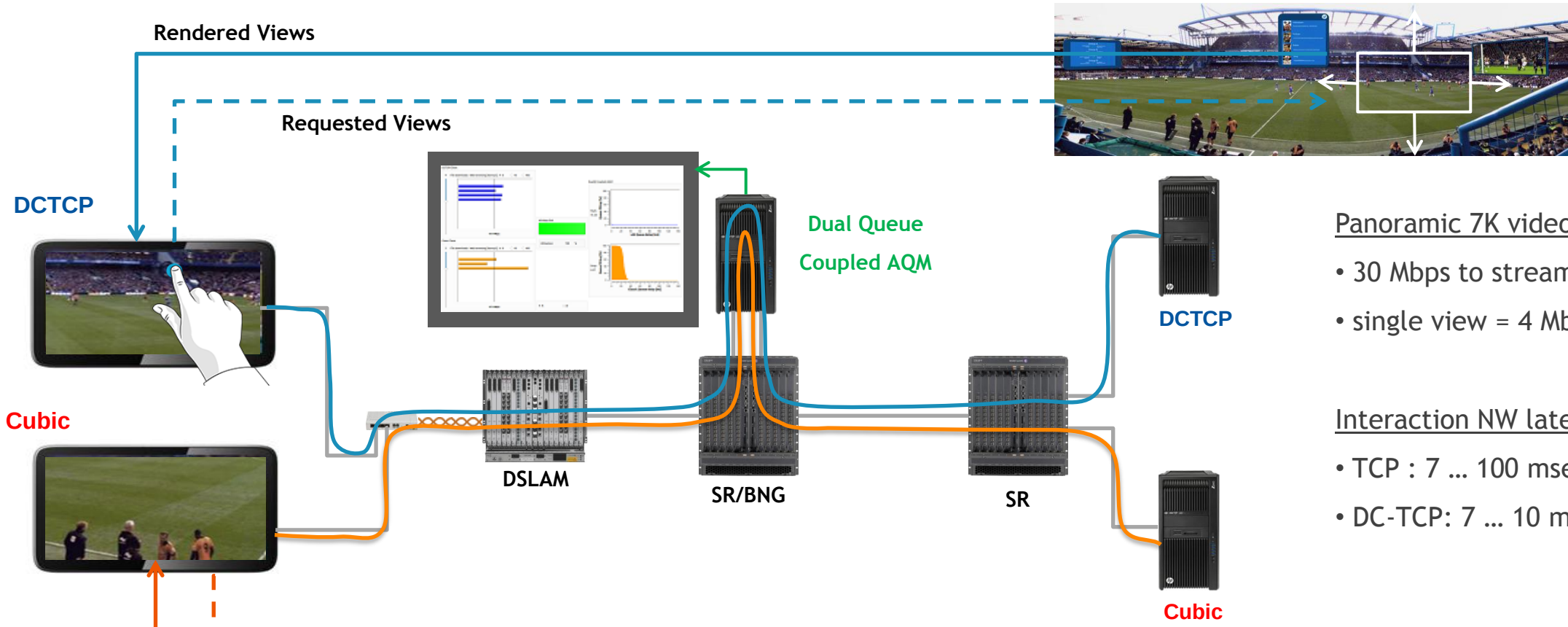
Conclusions

- Supports migration to new L4S family:
  - Low latency, Low (Zero) Loss, Scalable with Throughput
  - DCTCP behaves as an L4S, but evolution is possible
  - “Don’t think twice for L4S” solves coexistence with Classic TCPs (Reno, Cubic)
  - Unmanaged service deployment (no extra slices, weights or classes to be defined)
  - DualQ Coupled AQM provides similar and improved results as fq_codel with only 2 queues and no flow identification

- DCTCP should be made safer for Internet deployment
  - Controlled environments (Data Centres, Walled Garden) might need less safety

Demo at BitsNBytes

Example Application: Cloud-hosted Interactive Panoramic Video



Panoramic 7K video

- 30 Mbps to stream
- single view = 4 Mbps

Interaction NW latencies:

- TCP : 7 … 100 msec
- DC-TCP: 7 … 10 msec

R/TE
Reducing Internet Transport Latency

Questions

koen.de_schepper@alcatel-lucent.com

Backup

koen.de_schepper@alcatel-lucent.com

Opportunity to support a new TCP CC family

Classic Congestion Controller Family

- Reno with $r \approx 1/\sqrt{p}$

- X RTTs per drop event
  - for Reno: X=45 RTTs on 40Mbps 20ms
    X= 5600 RTTs on 1Gbps 100ms
    → gets worse in the future (also for Cubic)

- Designed for Drop based networks in the 80'
- Was known not to scale to higher throughputs

L4S Congestion Controller Family

- L4S with $r \approx 1/p$

- C marked packets per RTT
  - (C=2 for DCTCP)
    → frequent feedback is better control!

- Design now for the future, using ECN better, to provide low loss, low latency and scalability
- DCTCP is big step forward and can be improved with incremental evolution

THROUGHPUT RED*

D = DCTCP C = CUBIC

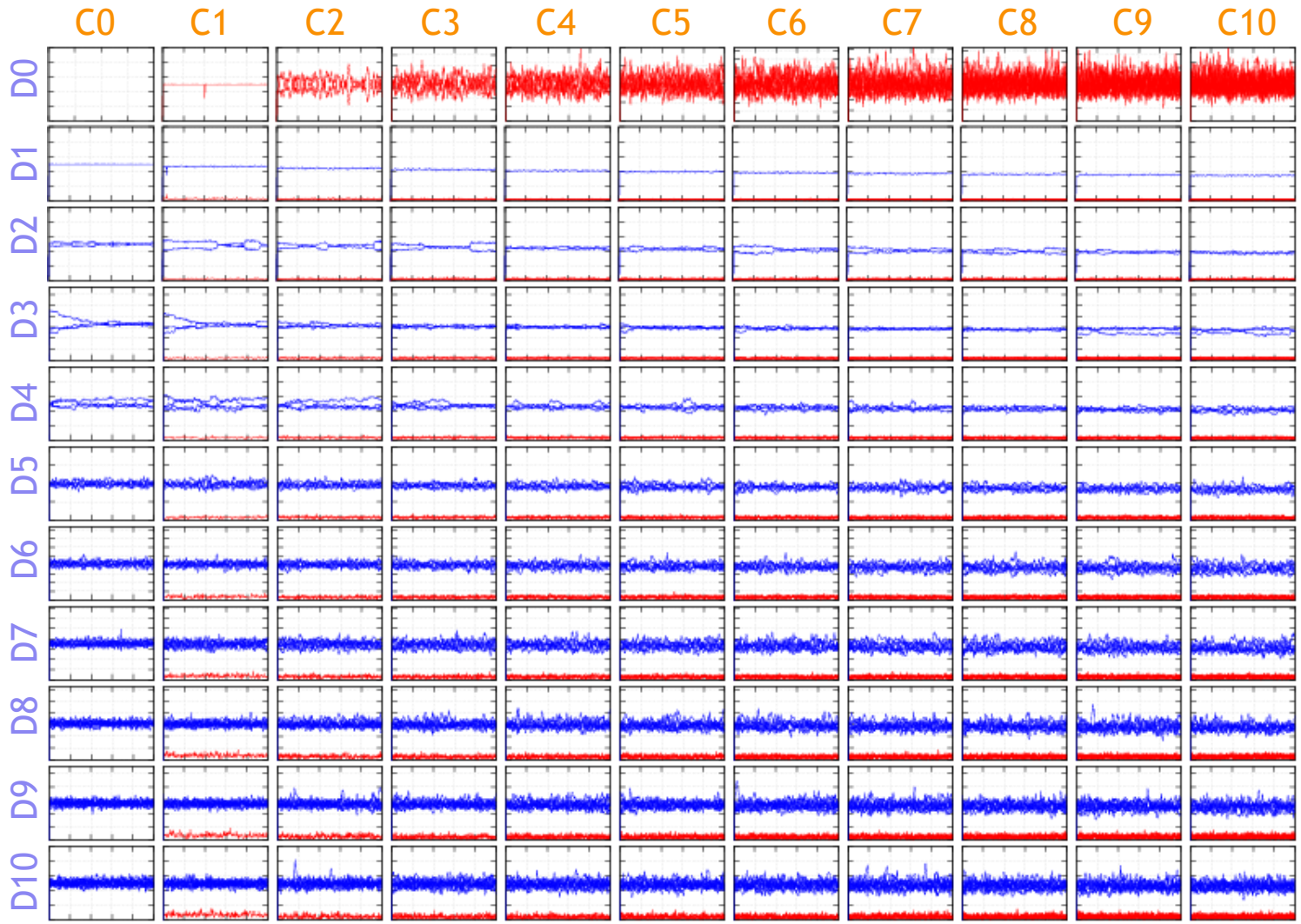
X = [0:250] sec

Y = [0:80/N] Mbps

first row: N = nbr of cubic

other rows: N = nbr of dctcp

→ No throughput equality: 40x less



R/TE
Reducing Internet Transport Latency

* tc qdisc add dev eth2 root red limit 1600000 min 120000 max 360000 avpkt 1000 burst 220 ecn bandwidth 40Mbit

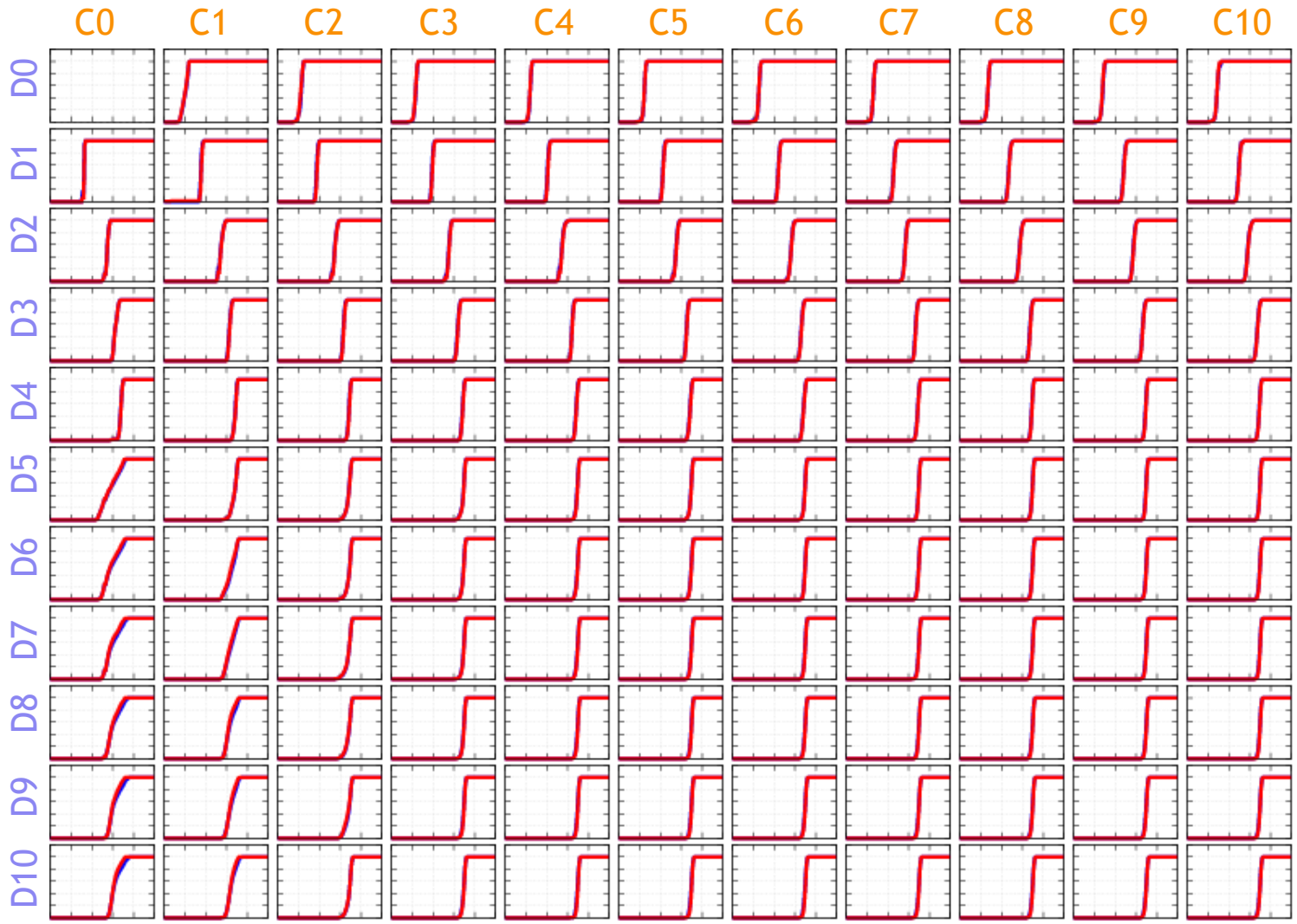
QSIZE CDF RED*

D = DCTCP C = CUBIC

X = [0:100] ms

Y = [0:120] %

→ Max queue size



* tc qdisc add dev eth2 root red limit 1600000 min 120000 max 360000 avpkt 1000 burst 220 ecn bandwidth 40Mbit

THROUGHPUT PIE*

D = DCTCP C = CUBIC

X = [0:250] sec

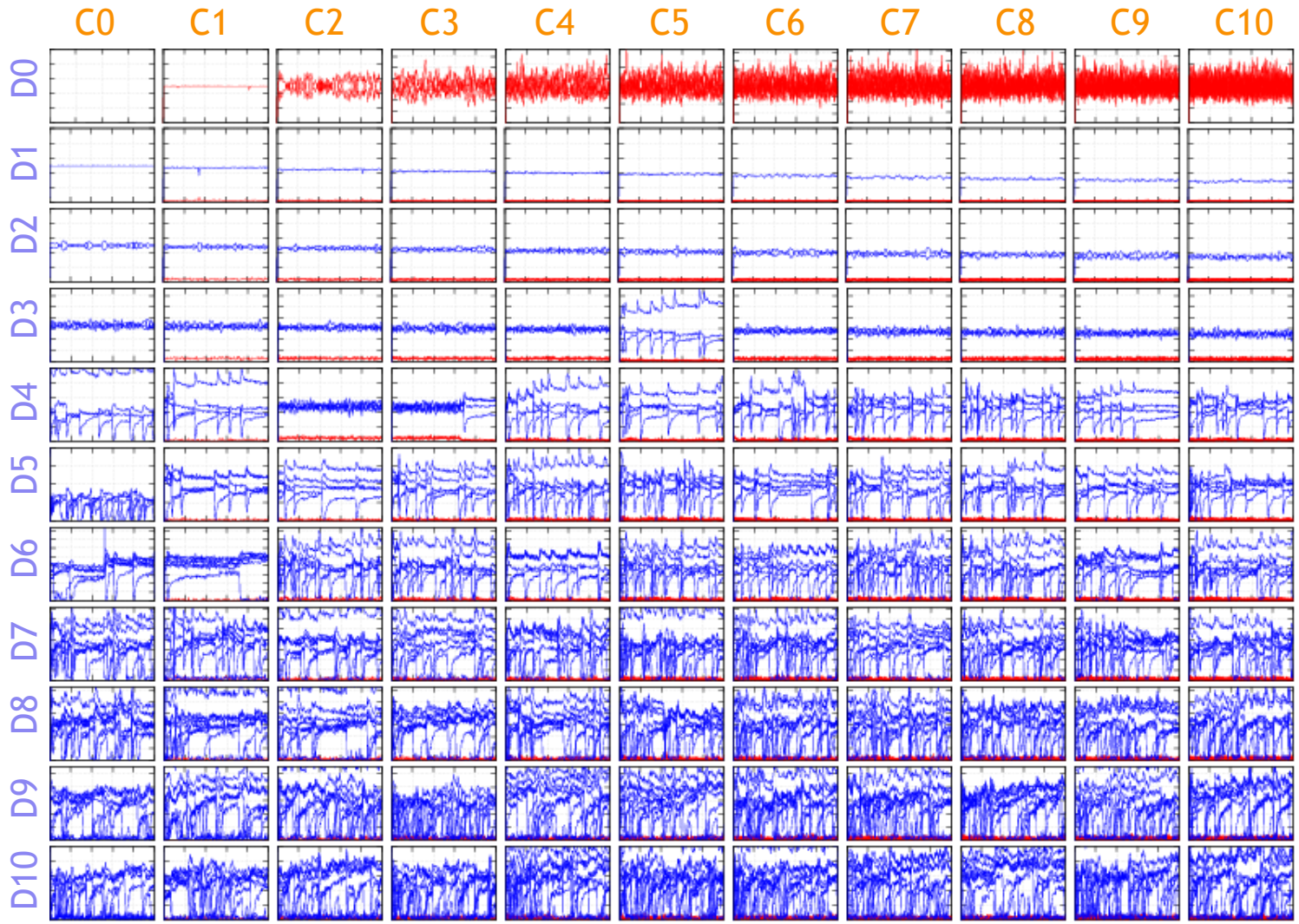
Y = [0:80/N] Mbps

first row: N = nbr of cubic

other rows: N = nbr of dctcp

→No throughput equality: 40x less

→Instable DCTCP throughput



* tc qdisc add dev eth2 root pie ecn

QSIZE CDF

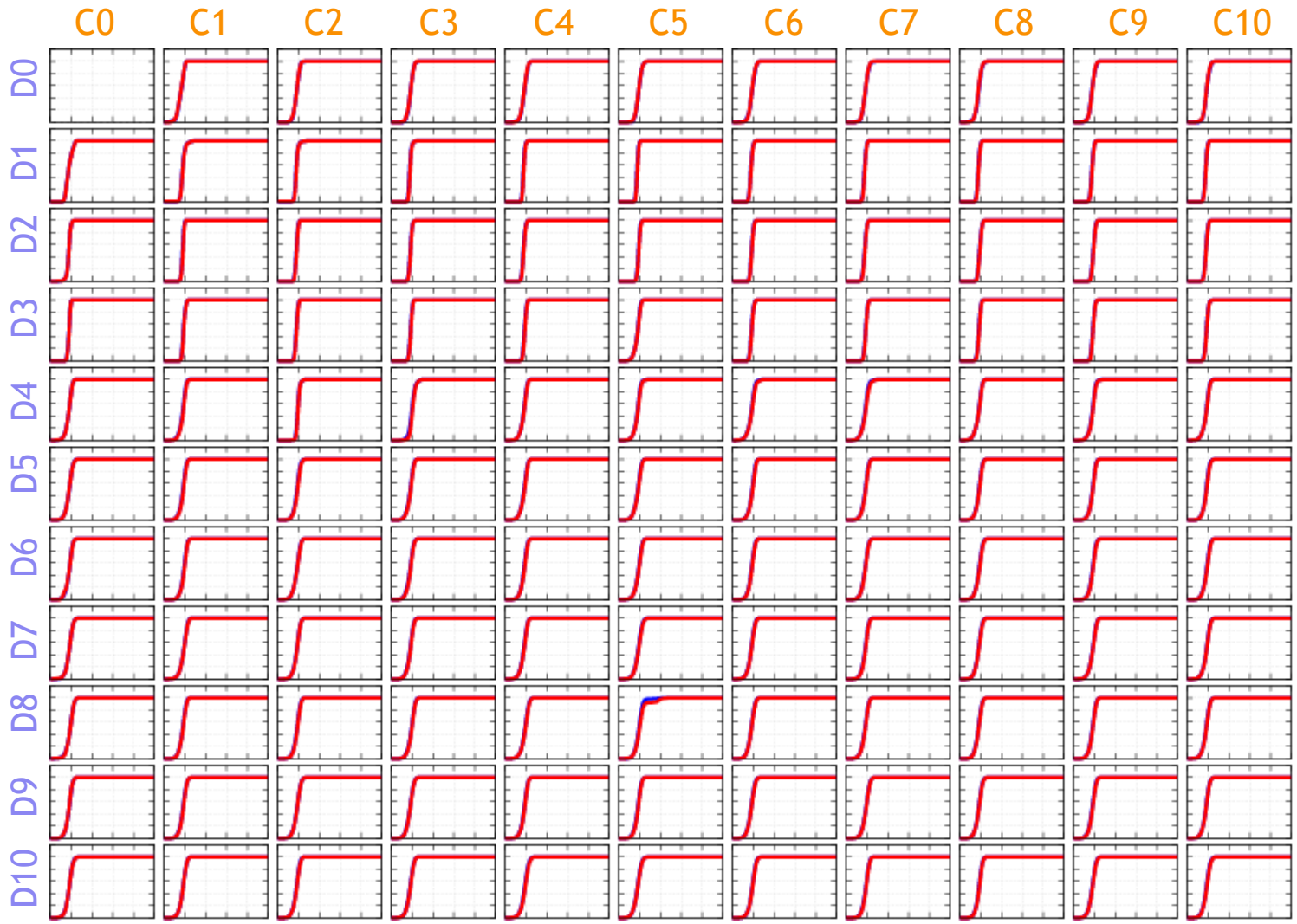
PIE*

D = DCTCP

C = CUBIC

X = [0:100] ms

Y = [0:120] %



→ PIE achieves target Q size

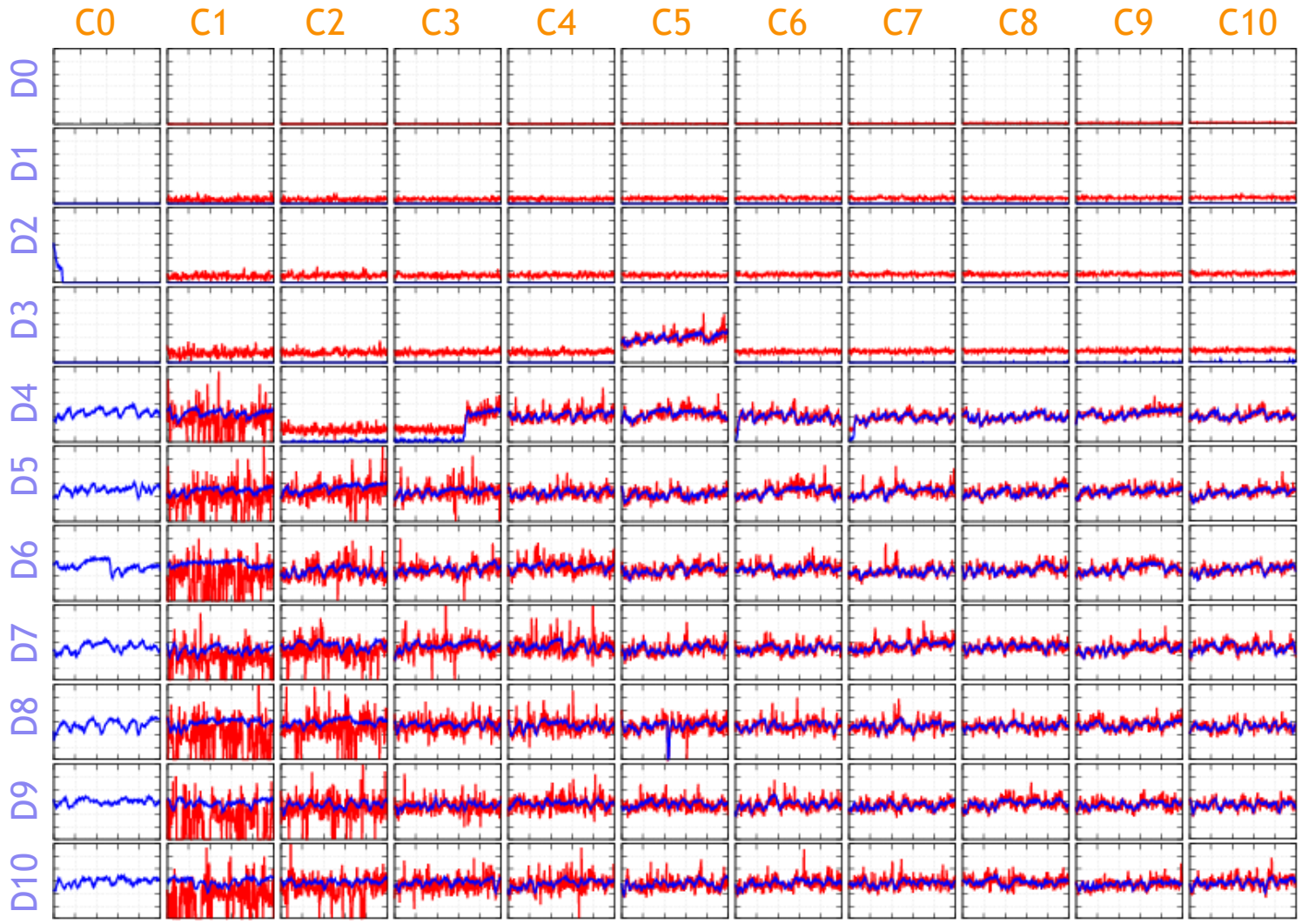
* tc qdisc add dev eth2 root pie ecn

DROP PROBABILITY PIE*

D = DCTCP C = CUBIC

X = [0:250] sec

Y = [0:60] %



→PIE only drops above 10%

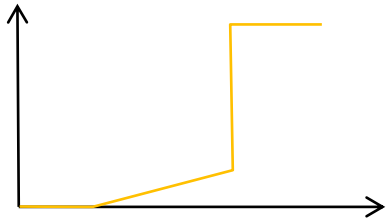
→DCTCP behaves ‘strange’

* tc qdisc add dev eth2 root pie ecn

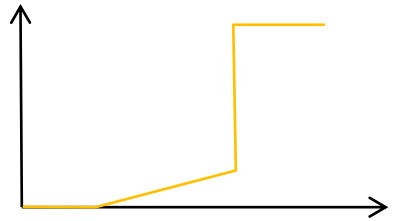
TCP CONGESTION CONTROL SCHEMES
STEADY STATE RATE

- Steady state rate has been calculated for existing CC schemes:

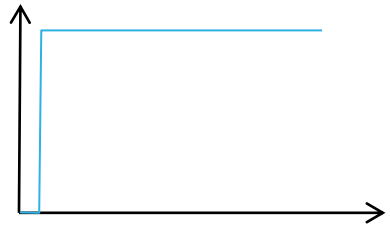
$$r_{reno} = \frac{1.22}{p^{1/2}\mathrm{RTT}}$$



$$r_{cubic} = \frac{1.17}{p^{3/4}\mathrm{RTT}^{1/4}}$$

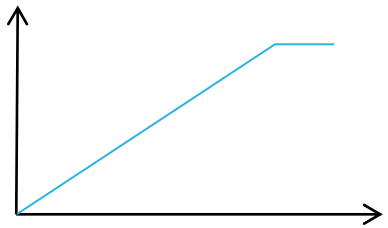


$$r_{dc} = \frac{2}{p^{2} \cdot \mathrm{RTT}}$$



- But we calculated that DCTCP running in non-on/off mode behaves as:

$$r_{dc\_p} = \frac{2}{p \cdot \mathrm{RTT}}$$



LOWER LATENCY BY SMARTER USE OF ECN
DATA CENTER TCP

TCP (Reno) ↔ **DCTCP**

Response to congestion in sender

- Half the congestion window when drop detected in one RTT
- Reduce partially per marked packet; half if all marked in one RTT

➔ React according to level of congestion

ECN feedback in receiver

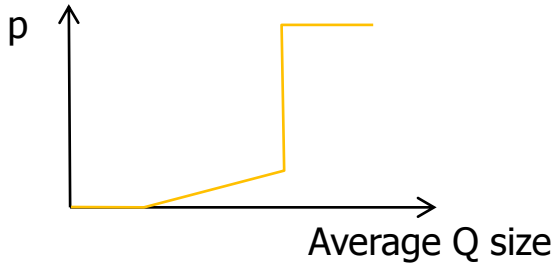
- Echo Congestion Experienced (CE) until sender acknowledges Congestion Window Reduced (CWR)
- Echo marking state of received packets without acknowledgement ➔ accurate ECN feedback

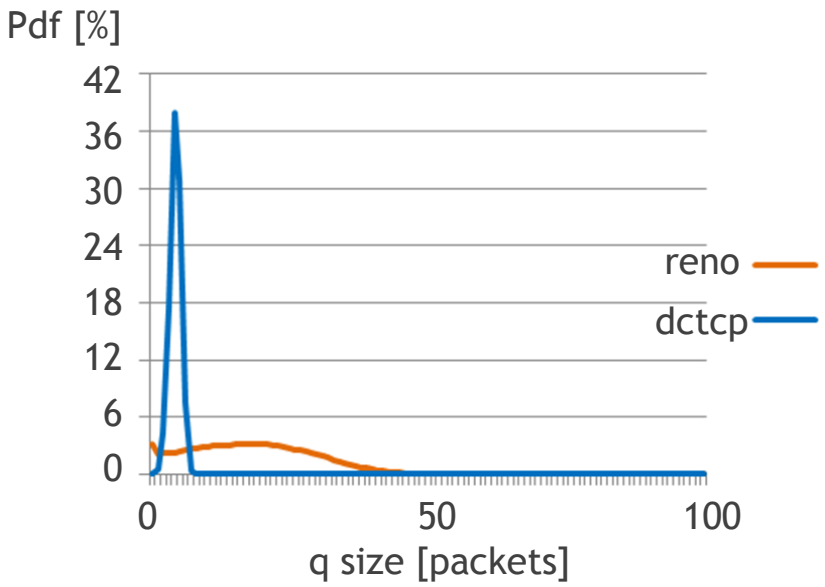
ECN marking in network

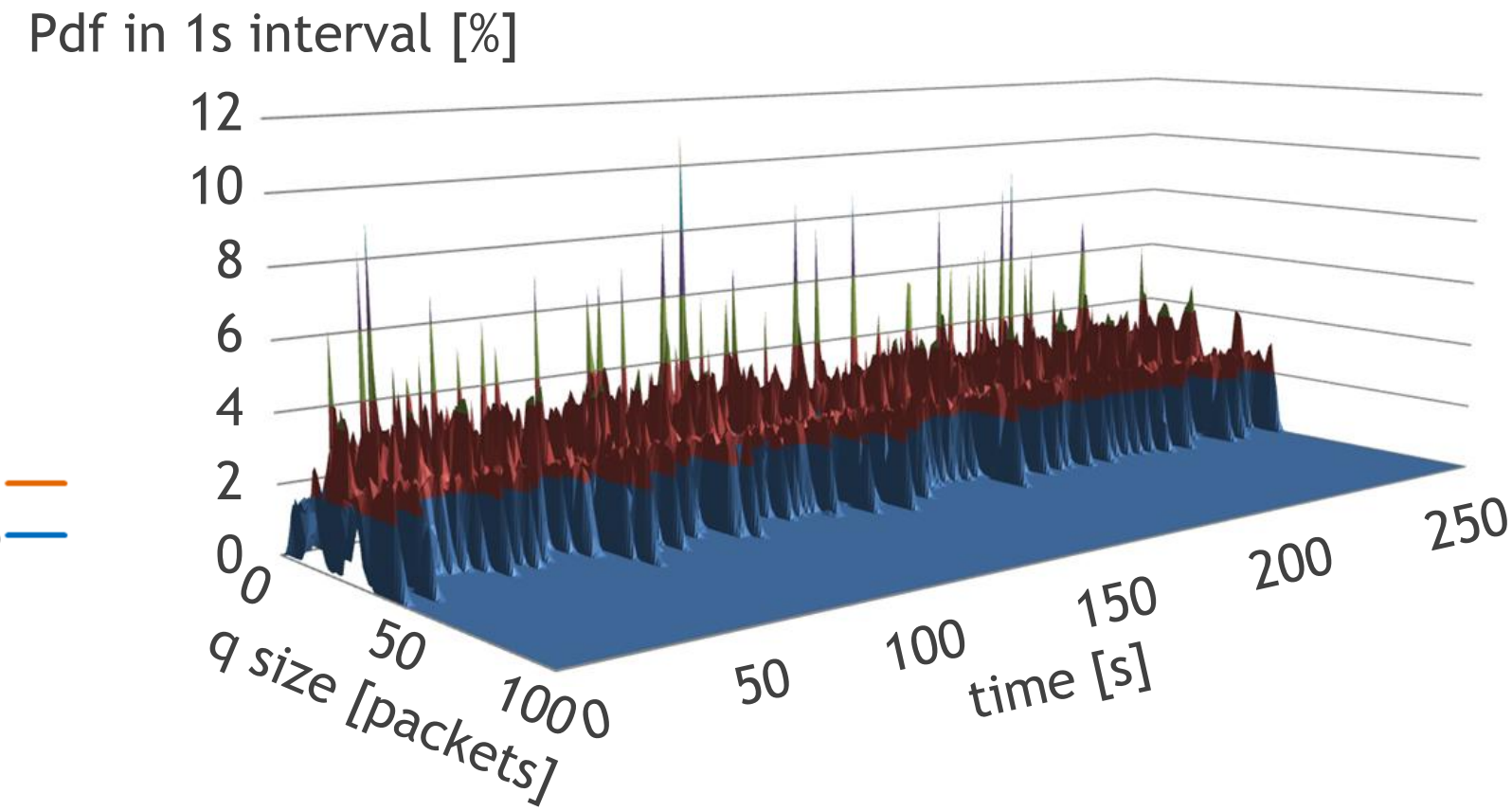
- Smooth and delay a drop or mark to allow bursts
- Don’t smooth or delay queue size
- Shallower marking threshold

➔ immediate ECN marking

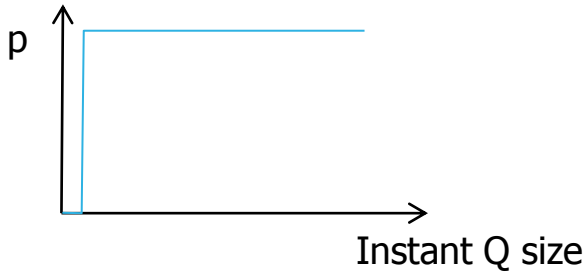
QUEUE SIZE AT DEQUEUE
1 TCP RENO FLOW (STEADY STATE)

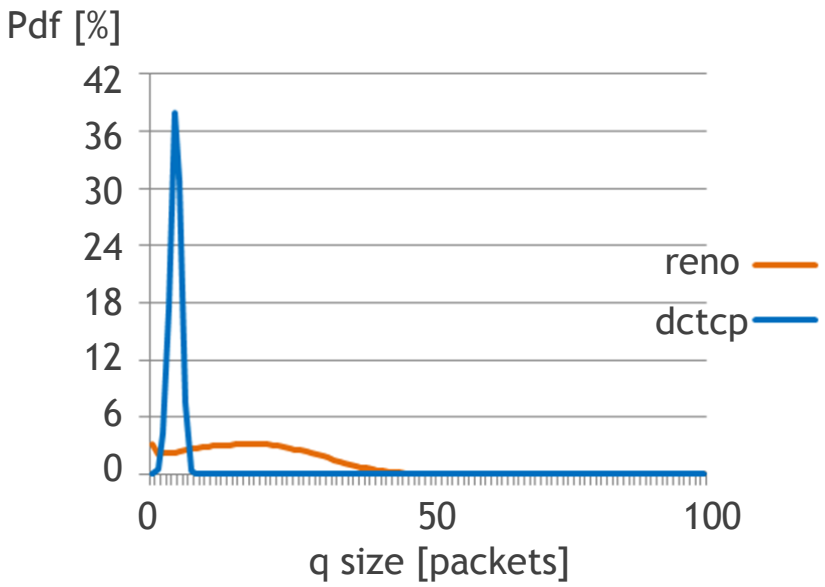


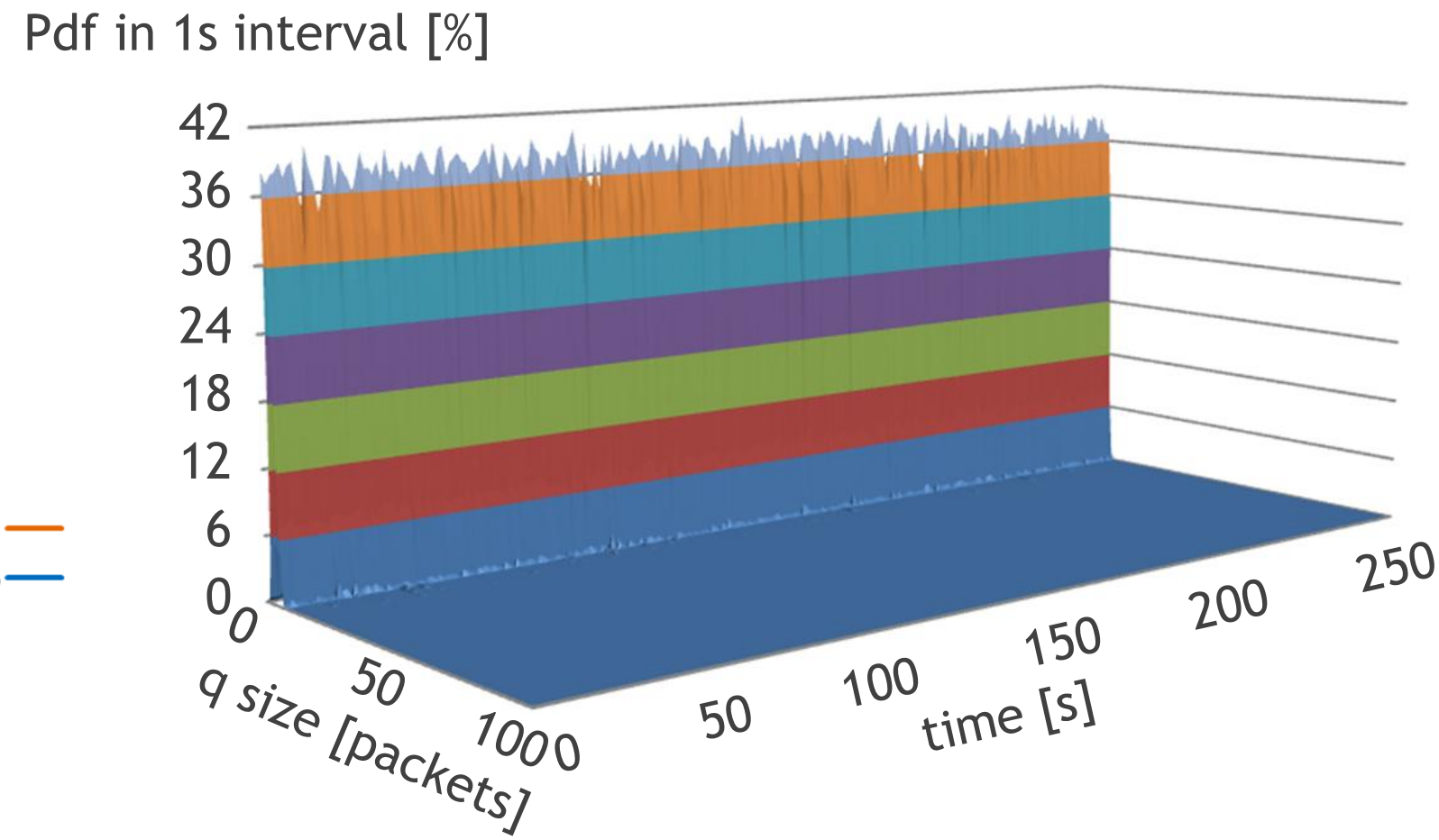




QUEUE SIZE AT DEQUEUE
1 DCTCP FLOW (STEADY STATE)







Scalable Congestion Controllers

Throughput Scalability:

Signal frequency (marked packets per RTT) is constant

$$r = \frac{C}{p.\mathrm{RTT}} \qquad r.p = \frac{C}{\mathrm{RTT}}$$

Congestion signal frequency:

C marked per RTT (C=2 for DCTCP)

→ frequent feedback is better control!

Classic Congestion Controllers

Compromise for drop:

Allows AQM to Think twice before dropping → $p_d = p^2$

$$r = \frac{C}{\sqrt{p_d}.\mathrm{RTT}} \qquad r.p_d = \frac{f(r,\mathrm{RTT})}{\mathrm{RTT}} \qquad f(r,\mathrm{RTT}) = \frac{C^2}{\mathrm{r.RTT}} = \frac{1}{X}$$

Congestion signal frequency:

X RTTs per drop (for Reno: X=45 RTTs on 40Mbps 20ms)
X= 5600 RTTs on 1Gbps 100ms)

→ gets worse in the future (also for Cubic)

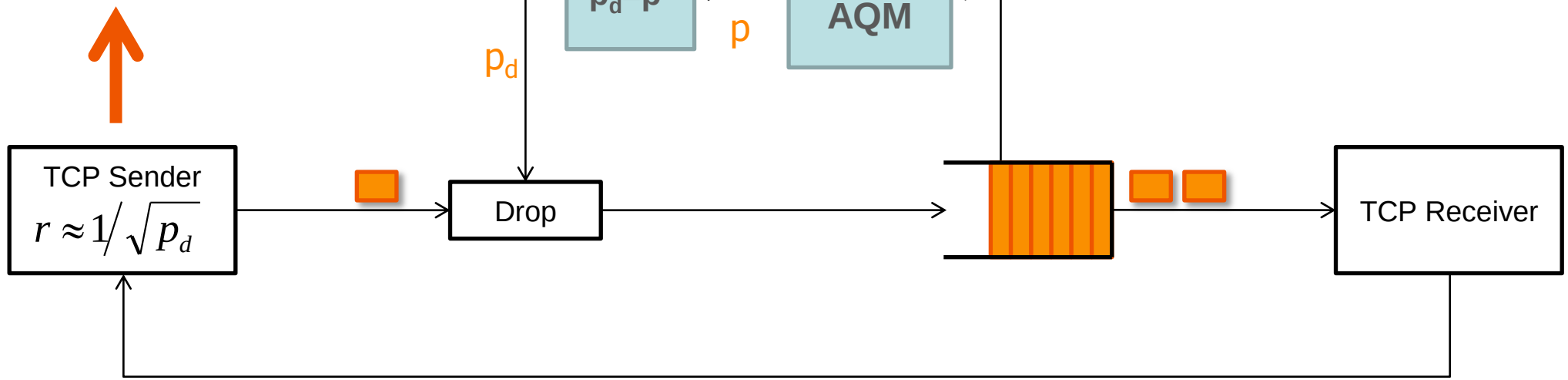
Linearity of thinking twice at the output of AQM

$$p_d \approx p^2$$

$$drop = p > \max(R_1 \,\&\, R_2)$$

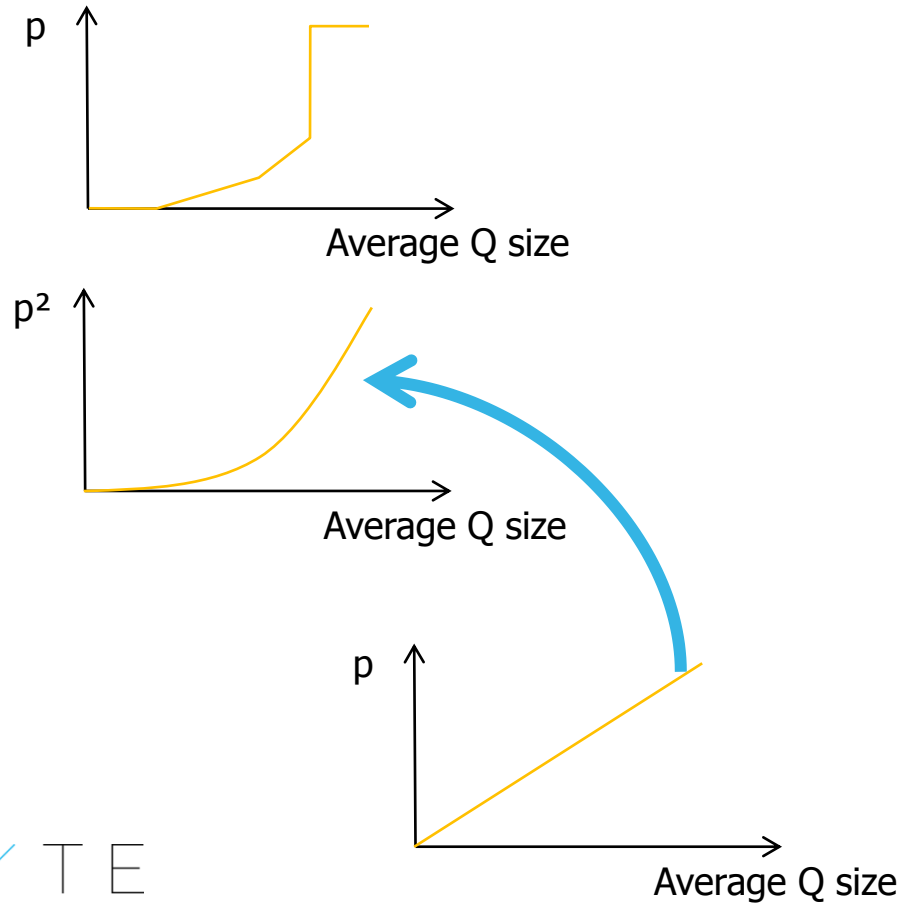
AQM can be simplified, as its output is linear proportional to congestion level

Can be controlled as a Scalable CC with a non-scalable (drop) signal

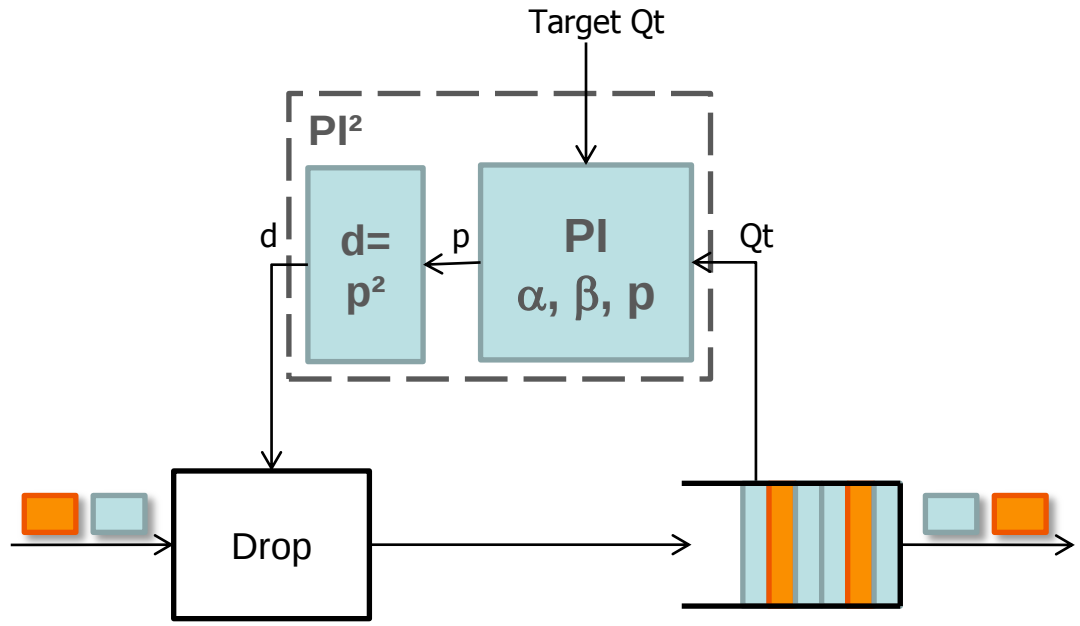


Linearity of thinking twice at the output of AQM

Simpler RED: Curvy RED



Simpler PIE: PI²



No need to autotune α, β to level of p²

p is linear with the congestion level

parameter	PIE			PI²		
Internal p	30,00%	3,00%	0,30%	54,77%	17,32%	5,48%
I-gain α	0,125	0,0625	0,015625	0,125	0,125	0,125
P-gain β	1,25	0,625	0,15625	1,25	1,25	1,25
ΔIntegral	**Δp by PIE**			**Δp by PI²**		
100ms	1,2500%	0,6250%	0,1562%	1,3849%	0,4486%	0,1525%
10ms	0,1250%	0,0625%	0,0156%	0,1370%	0,0434%	0,0138%
1ms	0,0125%	0,0062%	0,0015%	0,0136%	0,0043%	0,0013%
-1ms	-0,0125%	-0,0062%	-0,0015%	-0,0136%	-0,0043%	-0,0013%
-10ms	-0,1250%	-0,0625%	-0,0156%	-0,1367%	-0,0431%	-0,0135%
ΔPoportional	**Δp by PIE**			**Δp by PI²**		
100ms	12,5000%	6,2500%	1,5625%	15,2555%	5,8926%	2,9318%
10ms	1,2500%	0,6250%	0,1562%	1,3849%	0,4486%	0,1525%
1ms	0,1250%	0,0625%	0,0156%	0,1370%	0,0434%	0,0138%
-1ms	-0,1250%	-0,0625%	-0,0156%	-0,1367%	-0,0431%	-0,0135%
-10ms	-1,2500%	-0,6250%	-0,1562%	-1,3536%	-0,4173%	-0,1213%

Think Once to Mark, Twice to Drop
to align the throughput compatibility

Classic Congestion Controller Family

L4S Congestion Controller Family

$r \approx 1/\sqrt{p_d}$

$p_m \approx p$

$p_d \approx p^2$

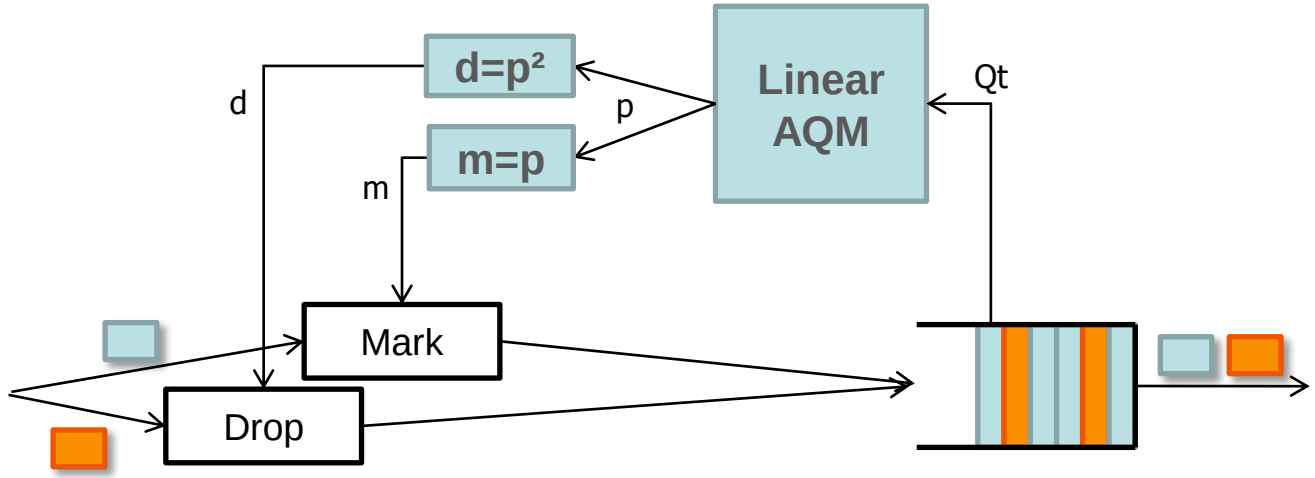
$r \approx 1/p_m$

$p_d \approx p^2$

$drop = p > \max(R_1 \,\&\, R_2)$

$p_m \approx p$

$mark = p > R_1$



$r_{scalable} \approx r_{classic}$

Can we still use ECN-bits in IP as Family indicator?