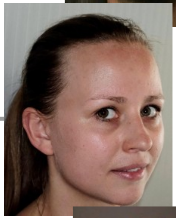
Implementing the 'Prague Requirements' in TCP for L4S

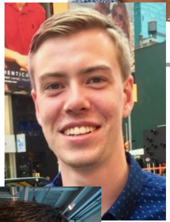
Bob Briscoe (Independent & CableLabs)

Koen De Schepper (Nokia Bell-Labs)

Olivier Tilmans (Nokia Bell-Labs)

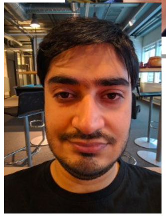


Mirja Kuehlewind (ETHZ)



Joakim Misund (Uni Oslo & Simula Research Lab)

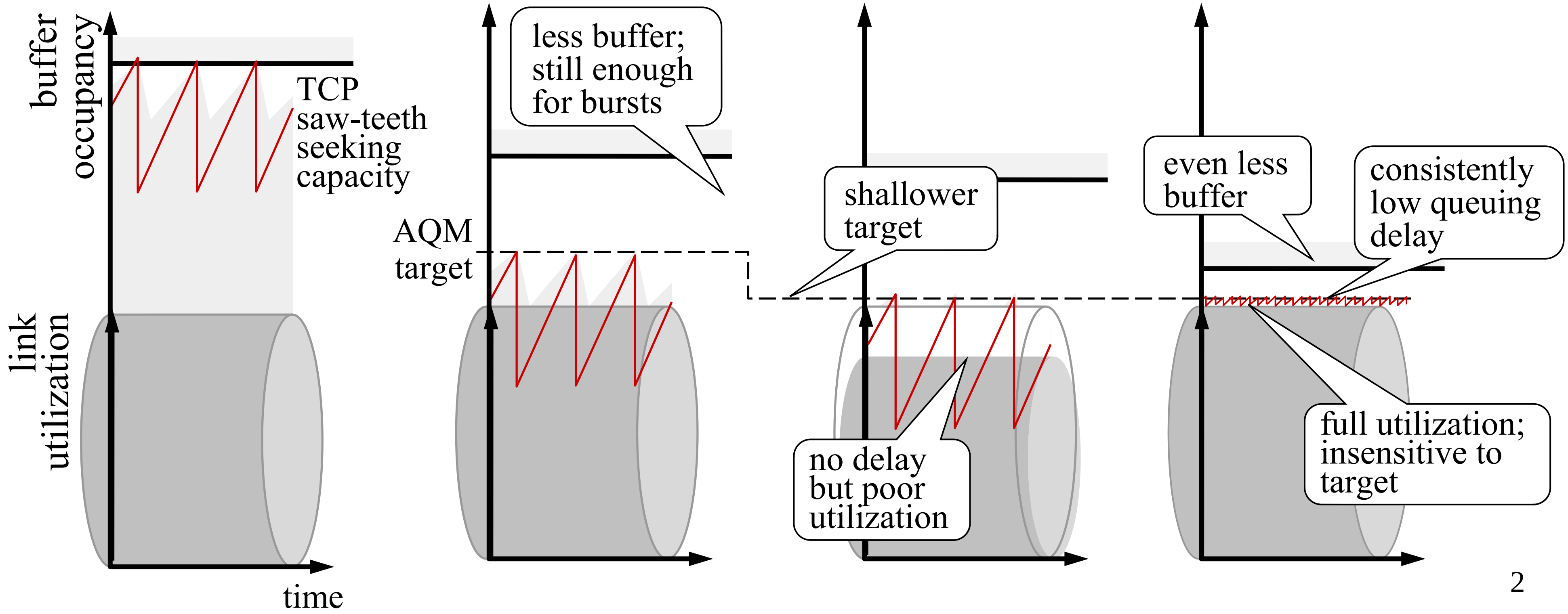
Olga Albisser née Bondarenko (Simula Research Lab)



Asad Sajjad Ahmed (Simula Research Lab & Uni Oslo)

The trick: scalable congestion control

	① Today (typical)	② Today (at best)	③ Unacceptable	④ L4S
Bottleneck	Bloated drop-tail buffer	AQM	Shallower AQM	Immediate AQM
Sender CC	Classic	Classic	Classic	Scalable (tiny saw-teeth)



Implementation status

pasted from https://riteproject.eu/dctth/#code

Source Code

- Dual Queue Coupled AQM
  - with PI2: Linux repo
  - With Curvy RED (TBA)
- TCP Prague
  - Linux repo
- QUIC Prague
  - General repo (should work for Linux, FreeBSD, Windows)
- SCReAM (Self-Clocked Rate Adaptation for Multimedia) a mobile optimised congestion control algorithm for real-time interactive media, with support for L4S
  - General repo

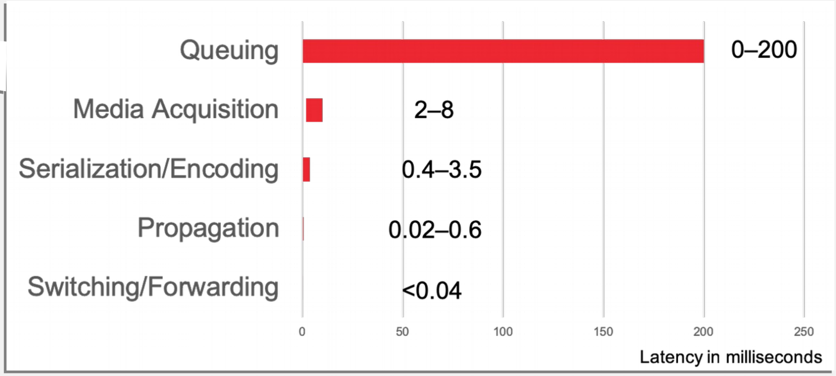
+DOCSIS 3.1
(next slide)

- Component parts
  - Accurate ECN TCP Feedback (included in TCP Prague above)
    - Linux repo and Linux repo without AccECN TCP Option
  - Paced Chirping
    - for Linux (initial proof-of-concept research code)
  - Data Centre TCP (DCTCP) for
    - Linux (in the mainline kernel)
    - FreeBSD (in the mainline kernel)
    - ns2 patch.

particular thanks to Olivier Tilmans
for pulling together TCP Prague and the Hackathon team

Low Latency DOCSIS 3.1

- Low Latency measures mandatory from Jan'19
  - upstream (Cable Modem) & downstream (CMTS)
    - DOCSIS 3.1 MAC and Upper Layer Protocols i/f (MULPI) Spec (i17+)
    - Cable Modem Operations Support System Interface Spec (i14+)
    - CCAP Operations Support System Interface Specification (i14+)
- Cuts 2 main sources of delay
    - MAC: Request-grant loop
    - Queuing: **Mandatory L4S support**
- White paper: Low Latency DOCSIS: Technology Overview
    - Also translated into ASCII: draft-white-tsvwg-lld (Informational)
- Certification test plans nearing completion
- Implementation in progress



TCP Prague

- new Linux congestion control module
  - with mandatory use of certain improvements to base TCP
- based off DCTCP scalable congestion controller
  - some improvements available for DCTCP as well
- usable for testing, but still a work in progress
  - Available from:
    https://github.com/L4STeam/tcp-prague (tcp_prague branch)
  - patch submitted to Linux netdev list this week
  - To load & enable
    sudo modprobe tcp_prague
    sudo sysctl -w net.ipv4.tcp_congestion_control=prague

The 'Prague L4S requirements'

- for scalable congestion ctrls over Internet
  - Assuming only partial deployment of either FQ or DualQ Coupled AQM isolation for L4S
  - Jul 2015 Prague IETF, ad hoc meeting of ~30 DCTCP folks
  - categorized as safety (mandatory) or performance (optional)
- not just for TCP
  - behaviour for any wire protocol (TCP, QUIC, RTP, etc)
- evolved into IETF conditions for setting ECT(1) in IP

Requirements
L4S-ECN Packet Identification: ECT(1)
Accurate ECN TCP feedback
Reno-friendly on loss
Reno-friendly if Classic ECN bottleneck
Reduce RTT dependence
Scale down to fractional window
Detecting loss in units of time
Optimizations
ECN-capable TCP control packets
Faster flow start
Faster than additive increase

[ietf-l4s-id]

TCP Prague: status against Prague L4S requirements

Linux code:	none	none (simulated)	research private	research opened	**RFC**	**mainline**

Requirements	base TCP	DCTCP	TCP Prague
L4S-ECN Packet Identification: ECT(1)		**module option**	**mod opt default**
Accurate ECN TCP feedback	**sysctl option**	**?**	**mandatory**
Reno-friendly on loss		**inherent?**	**inherent**
Reno-friendly if classic ECN bottleneck			TBA if nec.
Reduce RTT dependence			simulated
Scale down to fractional window	in progress		
Detecting loss in units of time	**default RACK**	**default RACK**	mandatory
Optimizations			
ECN-capable TCP control packets	**sysctl option off**	**default on**	**default off→on later**
Faster flow start	in progress		
Faster than additive increase		in progress	

Set ECT(1) in IP header

Codepoint	IP-ECN bits	Meaning
Not-ECT	00	Not ECN-Capable Transport
ECT(0)	10	Classic ECN-Capable Transport
ECT(1)	01	L4S ECN-Capable Transport
CE	11	Congestion Experienced

- the L4S packet identifier
  - IPv4 & IPv6
- Added module option to TCP Prague, and DCTCP could duplicate it
  - TCP Prague: might replace DCTCP in a private DC

    ```
    sudo modprobe tcp_prague prague_ect=X
    ```

  - DCTCP: private DC might need to control which codepoint[1]

    ```
    sudo modprobe tcp_dctcp dctcp_ect=X
    ```

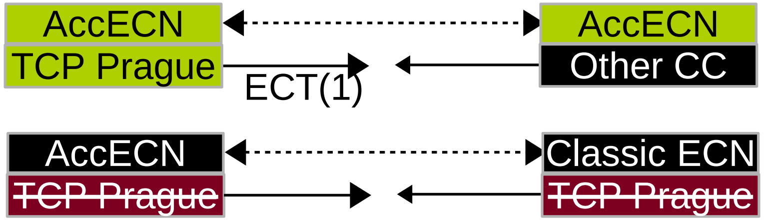
X		TCP Prague	DCTCP
0	use ECT(0)		default
1	use ECT(1)	default	

[1] only once the patch fixing dctcp's response to loss is accepted and response to loss is not disabled

Accurate ECN TCP feedback

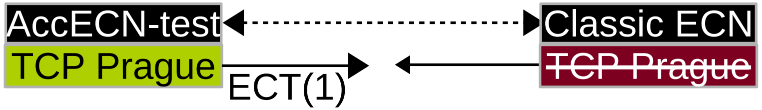
- TCP Prague, like DCTCP, needs extent of congestion, not just existence
  - Classic ECN [RFC3168] limits TCP feedback to one ECN-event per RTT
  - DCTCP redefines TCP flags, but no negotiation and unreliable

- AccECN: negotiated reliable f/b of extent of congestion
  - updates the base TCP stack – independent of TCP Prague
  - but TCP Prague depends on both ends having negotiated AccECN



- For testing, if AccECN negotiation fails, can force a classic ECN peer to echo each ECN mark once
  - not for production – unreliable delivery of congestion feedback

```
sudo sysctl -w net.ipv4.tcp_force_peer_unreliable_ece=1
```



see Mirja's netdev 2.2 talk: AccECN

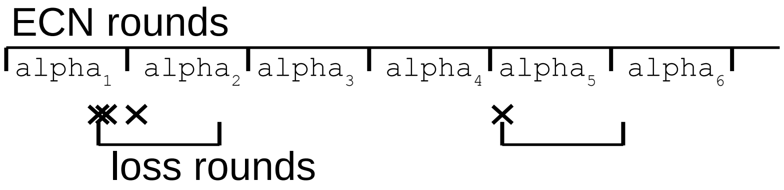
[RFC7560], [ietf-AccECN], [AccECN-netdev]

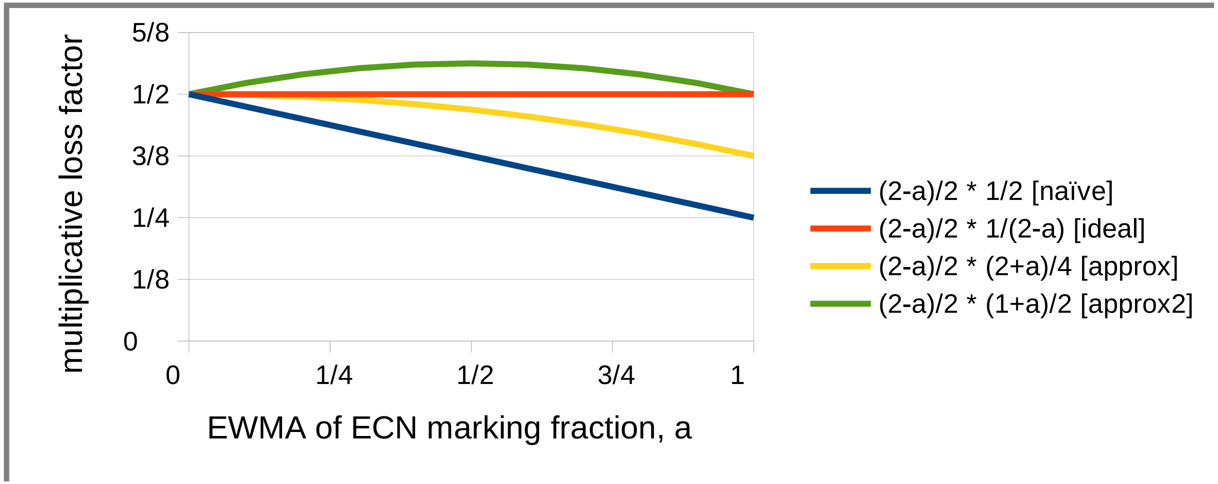
Fall back to Reno-Friendly on loss

- Linux DCTCP bug for last 4 years had no ssthresh reduction on fast re-xmt
    - fixed in TCP Prague, and patch submitted for DCTCP
- Multiplicative decreases of ECN & loss in 'same' RTT?
    - EWMA of ECN, `alpha` continues independent of losses
    - each response to a loss episode contrived
      to give a compound decrease of ~1/2

(ECN reduction) * (loss reduction) ≈ 1/2

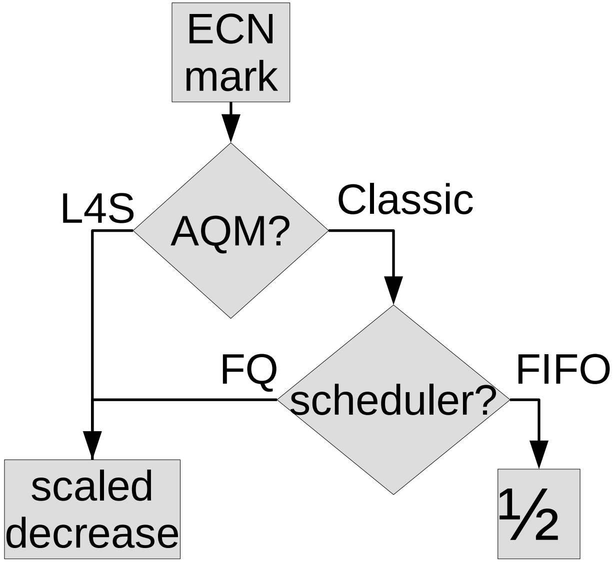
(2-alpha)/2 * `(2+alpha)/4` ≈ 1/2





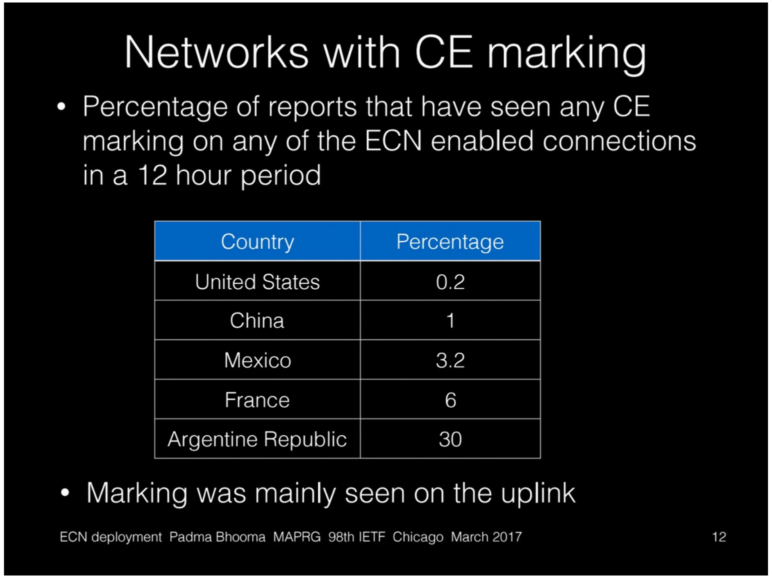
[RFC8257]

Fall back to Reno-Friendly on Classic ECN



- All academic ECN studies over the years (incl. 2017, 2019) have found virtually no ECN marking
- Would expect to see CE marking as FQ-CoDel deploys

- Are any FIFO Classic ECN routers enabled?
  - Mar 2017: Apple found at least 1 CE /12hr
  - FQ or FIFO?
    - Digging into Apple data
    - Devised FQ v FIFO test



Networks with CE marking

- Percentage of reports that have seen any CE marking on any of the ECN enabled connections in a 12 hour period

Country	Percentage
United States	0.2
China	1
Mexico	3.2
France	6
Argentine Republic	30

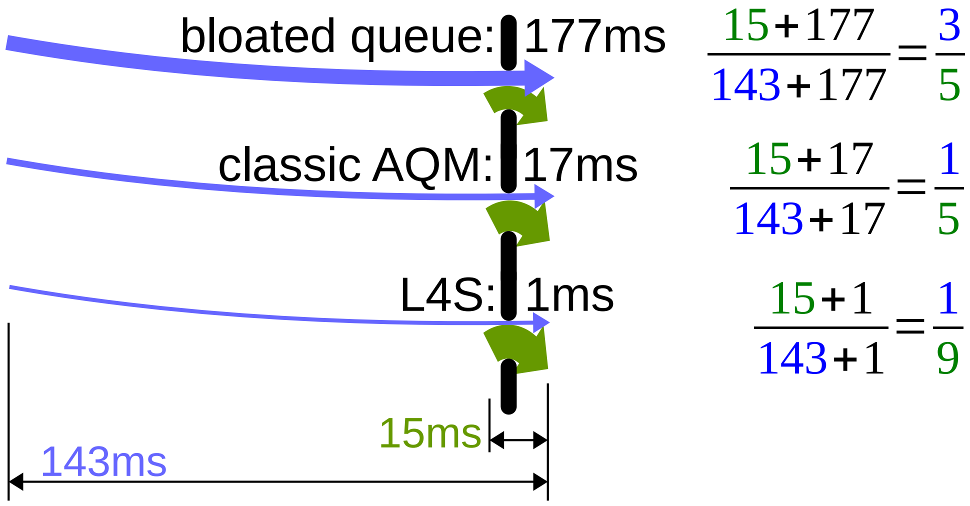
- Marking was mainly seen on the uplink

ECN deployment Padma Bhooma MAPRG 98th IETF Chicago March 2017 12

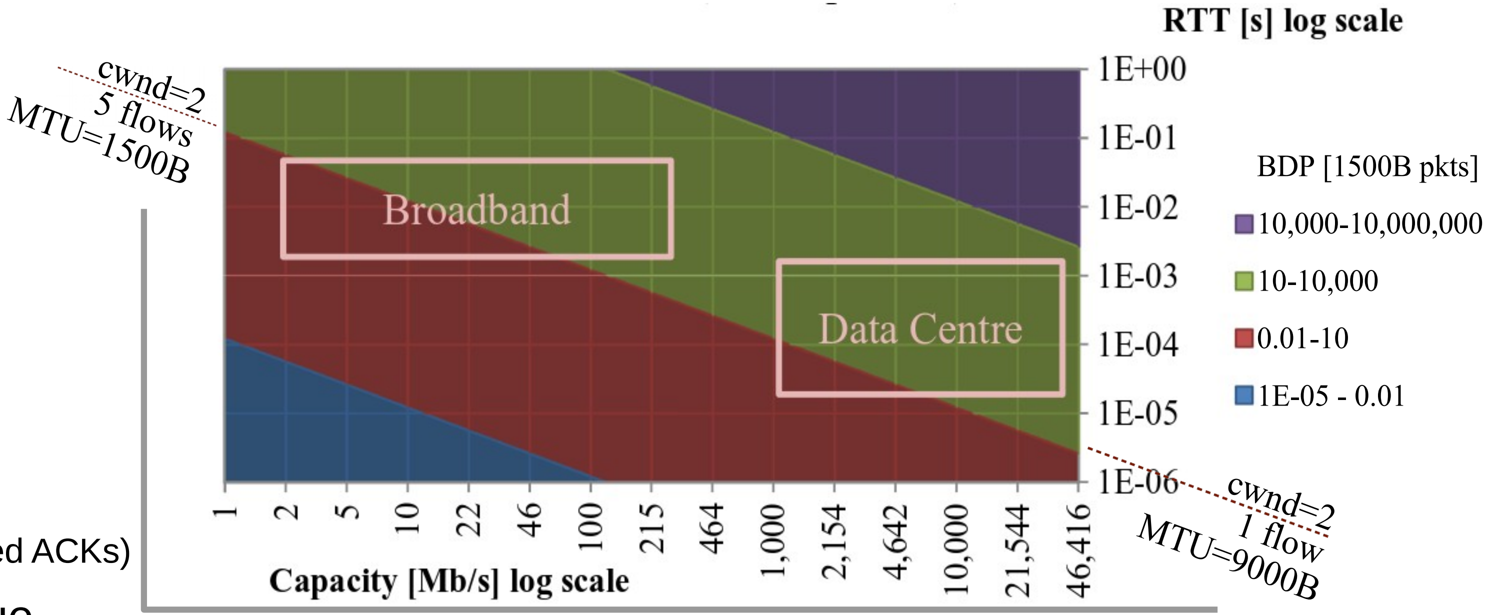
[RFC8257]

Reduce RTT Dependence

- Why worry?
  1) all TCP's are RTT-dependent anyway
  2) large & small RTT flow in one bottleneck is doubly rare
     - multiple long-running flows in one bottleneck are rare
     - large RTT flows are rare
- Because
  - RTT dependence only appears as queues shrink
  - today's rare traffic mix could be tomorrow's killer app
- Solutions simulated
  - not yet implemented in Linux



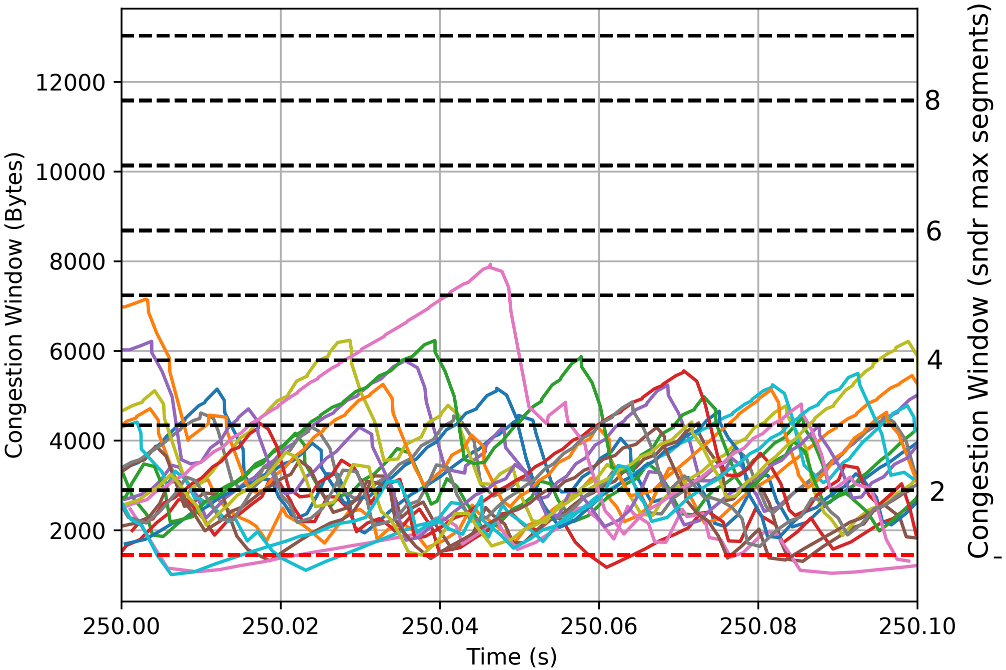
Scale down to fractional window



Problem

- TCP's min window:
  - 2 segments (due to delayed ACKs)
- with deep queue: no issue
- with shallow AQM target:
  - TCP becomes unresponsive
  - inflates queue to keep 2 pkts in flight / flow
  - ignores AQM's attempts to hold down queue

Scale down to fractional window



Solution

- fractional window implemented in base TCP
  - uses pacing (with ACK clock) for packet conservation
  - keeps AI below MD by scaling to log of ssthresh
- still debugging DCTCP-specific addition:
  - to adjust clocking of DCTCP EWMA
- once mature, will be added to TCP Prague
  - initially as default-off option

- testbed config of above example
  - 20 TCP Reno flows, SMSS 1448B
  - bottleneck link rate: 200Mb/s; base RTT 300µs
  - AQM: RED 0-10% over 1-3ms

[low-RTT-scaling], [CC-scaling-tensions]

Detecting loss in units of time

- like RACK (Recent ACKnowledgements)
- mandatory for L4S
  - L4S queues will know *all* sources tolerate reordering < x μs
  - softens a hard constraint on hi-speed link design
- TCP Prague implementation currently uses RACK as-is
  - including RACK's 3 DupACK bootstrap at start of flow – in units of packets :(
  - alternative approaches to modify RACK bootstrap for TCP Prague
    - bootstrap with SRTT/8 (say)
      using SRTT from 3WHS, dst cache and/or TFO cookie
    - ensure initial window is paced over the RTT,
      so duration of 3 DupACK is invariant as flow-rate scales

[RACK-netdev], [ietf-RACK]

Set IP-ECN to ECT on TCP control packets (ECN++)

- Loss protection for performance-critical pkts
- Already default-on in DCTCP
  - otherwise high background ECN-marking leads to high loss of not-ECT control packets
- Some fall-backs in IETF ECN++ spec still TBA
  - ECN++ default-off while fall-back code matures
  - to enable:

    `sudo sysctl -w net.ipv4.tcp_ecn_plus_plus=1`
- ECN++ patches the base TCP stack
  - TCP Prague depends on negotiating AccECN
  - So ECT(1) is allowed on every packet

TCP packet type	RFC3168 sender	ECN++ sender	
		AccECN f/b negotiated[1]	RFC3168 f/b negotiated
SYN	not-ECT	ECT	not-ECT
SYN-ACK	not-ECT	ECT	ECT
Pure ACK	not-ECT	ECT	not-ECT
Window probe	not-ECT	ECT	ECT
FIN	not-ECT	ECT	ECT
RST	not-ECT	ECT	ECT
Re-XMT	not-ECT	ECT	ECT
Data	ECT	ECT	ECT

[1] AccECN feedback 'requested' in the case of a SYN

'ECT' means whichever ECN-Capable Transport codepoint is appropriate, whether ECT(0) or ECT(1)

[ietf-ECN++]

Over-strict ECN negotiation 'bug'

- If a SYN requests ECN at the TCP layer and is already ECN-capable at the IP layer
  - Linux TCP listeners currently disable ECN for the connection
- Rationale: RFC3168 says “A host MUST NOT set ECT on SYN...”
  - so ECT on a SYN could be a symptom of network mangling
  - 'bug': it could also be a symptom of a new standard (it is now: ECN++)
- ECN++ client deployment cannot get started – it disables ECN :(

- Recent tiny patch for back-porting to all TCP listeners
  - identifies an ECN set-up SYN that's ECN-capable in IP by:

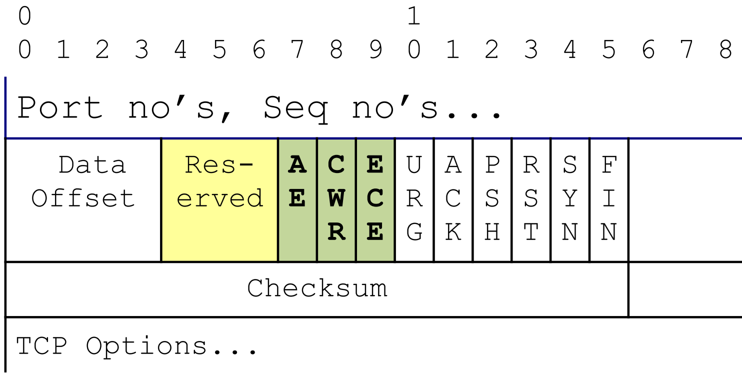
    ```
    flag bits 4-9 == 0b000011
    ```

    not just

    ```
    flag bits 8-9 ==     0b11
    ```

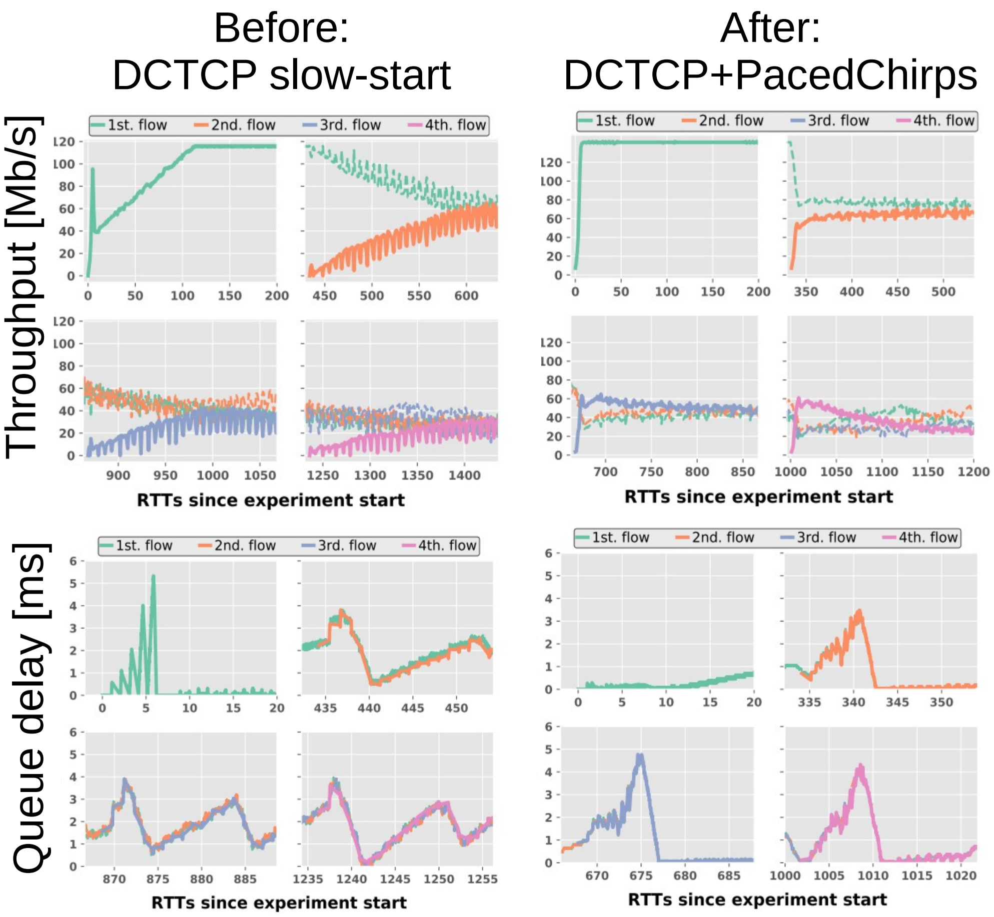
  - This can distinguish an RFC3168 ECN setup SYN from something newer that allows ECT on a SYN, such as an AccECN setup SYN, which uses

    ```
    flag bits 4-9 ==    0b111
    ```



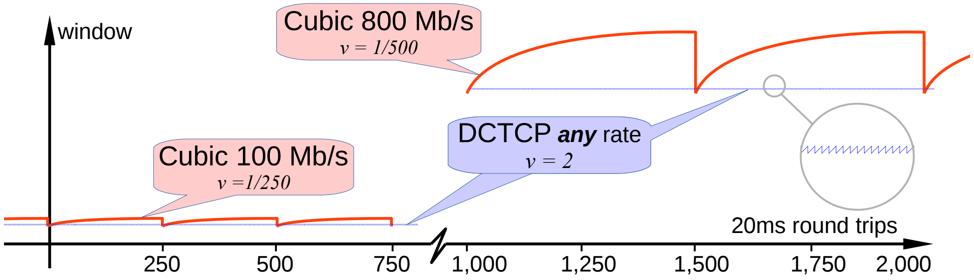
Faster flow start without overshoot

- DCTCP is slow to get started
  - <u>alone</u>: hard not to exceed the shallow ECN threshold
  - <u>pushing in</u>: frequent ECN marks hit new flow early
  - early exit from slow start into additive increase
- Why not respond to extent not existence?
  - TCP Prague sender doesn't know if marks are L4S
- Plan to use paced chirping, once stable
  - only a few packets of overshoot
  - typically much faster increase than slow-start
  - primarily delay-based, so universal
- Until paced-chirping is more mature
  - download separately: github.com/JoakimMisund/PacedChirping



Faster than additive increase

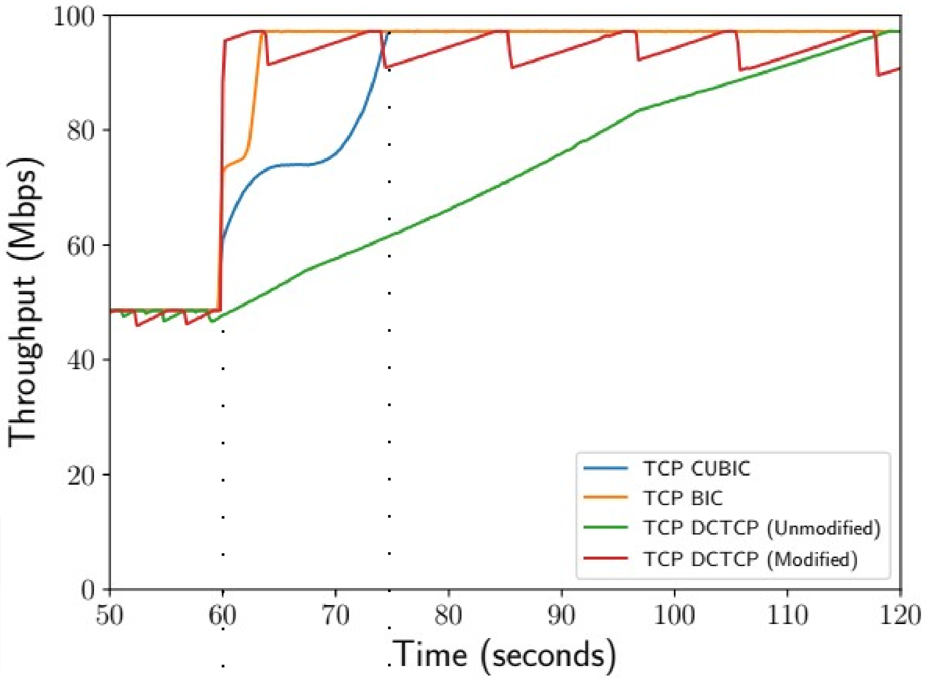
- DCTCP and TCP Prague can exploit high freq ECN
  - rapidly detect when it stops...
- ...then probe for capacity
  - plan to use paced chirping
- Why not use paced chirping all the time (like TCP Rapid)?
  - tried that – unnecessarily raises the noise floor

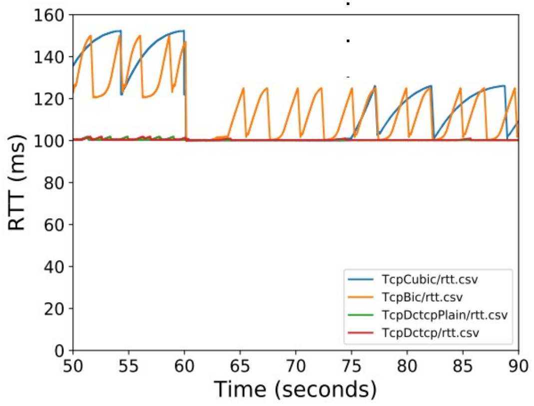


Faster than AI

- NS-3 simulation
  - Base RTT: 100ms
  - Capacity 50Mbps → 100Mbps

	RTTs to 95%	Overshoot
Cubic	150	20ms
DCTCP + Pc'd Chirping	7	<1ms



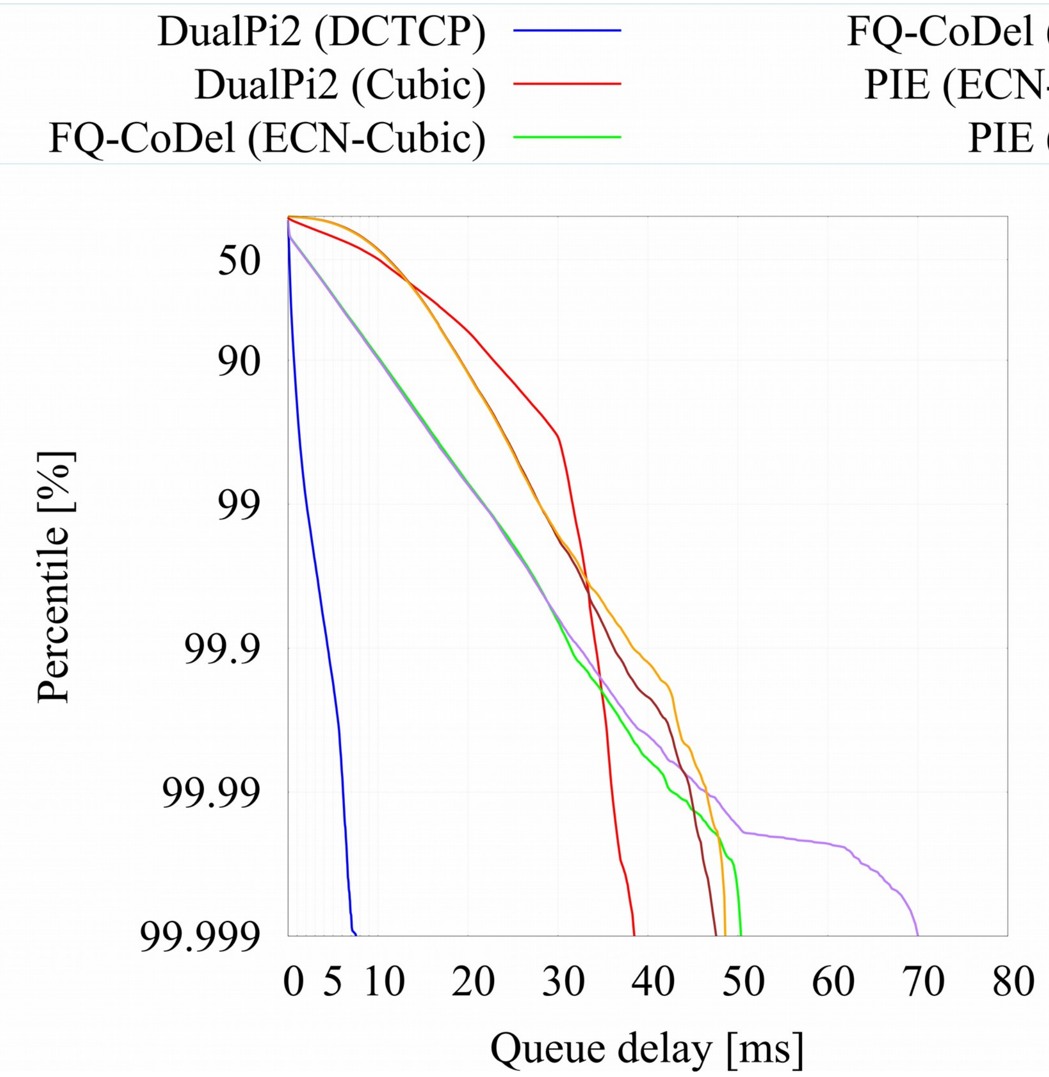


Prague L4S requirements: status

Linux code:	none	none (simulated)	research private	research opened	RFC	mainline

Requirements	base TCP	DCTCP	TCP Prague
L4S-ECN Packet Identification: ECT(1)		**module option**	**mod opt default**
Accurate ECN TCP feedback	**sysctl option**	**?**	**mandatory**
Reno-friendly on loss		**inherent?**	**inherent**
Reno-friendly if classic ECN bottleneck			TBA if nec.
Reduce RTT dependence			simulated
Scale down to fractional window	in progress		
Detecting loss in units of time	**default RACK**	**default RACK**	mandatory
Optimizations			
ECN-capable TCP control packets	**sysctl option off**	**default on**	**default off→on later**
Faster flow start	in progress		
Faster than additive increase		in progress	

Performance
(briefly)

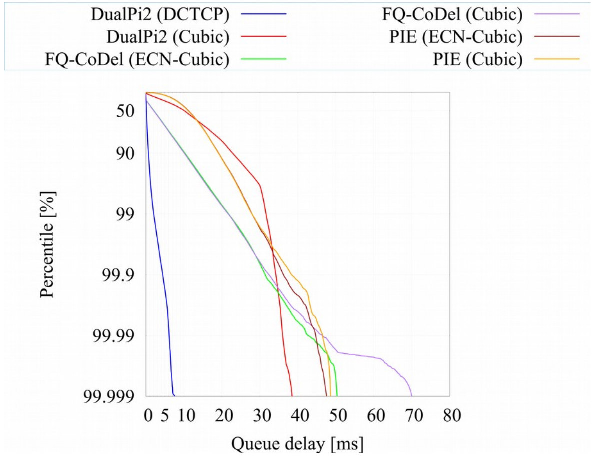


- Low higher percentile delay important
  - for low latency real-time delivery
  - have to measure delay of every packet

- median Q delay: 100-200μs
- 99%ile Q delay: 1-2ms
- **~10x lower delay than best 2nd gen. AQM**
  - at all percentiles
- ...when really hammering each AQM
  - long-running TCPs: 1 ECN 1 non-ECN
  - web-like flows @ 300/s ECN, 300/s non-ECN
    - exponential arrival process
    - file sizes Pareto distr. α=0.9 1KB min 1MB max
  - 120Mb/s 10ms base RTT

summary

- L4S
  - Frequent L4S-ECN markings give leaps in performance
    - Low Latency
    - Low Loss
    - Scalable throughput
  - a set of incremental changes to network & hosts
- TCP Prague
  - a set of incremental changes to TCP or DCTCP
  - with radical results
  - most are desirable in themselves



Requirements	base TCP	DCTCP	TCP Prague
Set ECT(1)		module option	mod opt default
Accurate ECN TCP feedback	sysctl option	?	mandatory
Reno-friendly on loss		inherent?	inherent
Reno-friendly if Classic ECN bottleneck			TBA if nec.
Reduce RTT dependence			simulated
Scale down to fractional window	in progress		
Measure reordering tolerance in time units	default RACK	default RACK	mandatory
Optimizations			
Set ECT in TCP control packets	sysctl option off	default on	default off→on later
Faster flow start	in progress		
Faster than additive increase		in progress	

more info

All via L4S Landing page: https://riteproject.eu/dctth/

Linux Netdev

- [tcp-prague-netdev] Briscoe, B., De Schepper, K., Albisser, O., Misund, J., Tilmans, O., Kühlewind, M. & Ahmed, A. S., “Implementing the 'TCP Prague' Requirements for L4S” in Proc. Netdev 0x13 (Mar 2019)
- [dualpi2-netdev] “DUALPI2 - Low Latency, Low Loss and Scalable (L4S) AQM” in Proc. Netdev 0x13 (Mar 2019)
- [paced-chirping-netdev] “Paced Chirping - Rethinking TCP start-up” in Proc. Netdev 0x13 (Mar 2019)
- [AccECN-netdev] Mirja Kühlewind, “State of ECN and improving congestion feedback with AccECN in Linux” in Proc. Netdev 2.2 (Dec 2017)
- [RACK-netdev] Cheng, Y. & Cardwell, N.,”Making Linux TCP Fast” in Proc. Netdev 1.2 (Oct 2016)

IETF

- [ietf-l4s-arch] Briscoe (Ed.), B., De Schepper, K. & Bagnulo, M., "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture," IETF Internet Draft draft-ietf-tsvwg-l4s-arch-03 (Oct 2018) (Work in Progress)
- [RFC8311] Black, D. “Explicit Congestion Notification (ECN) Experimentation” IETF RFC8311 (Jan 2018)
- [ietf-l4s-id] De Schepper, K., Briscoe (Ed.), B. & Tsang, I.-J., "Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay (L4S)," IETF Internet Draft draft-ietf-tsvwg-ecn-l4s-id-05 (Nov 2018) (Work in Progress)
- [RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L. & Judd, G., "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers," RFC Editor RFC8257 (October 2017)
- [ietf-dualq-aqm] De Schepper, K., Briscoe (Ed.), B., Albisser, O. & Tsang, I.-J., "DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput," IETF Internet Draft draft-ietf-tsvwg-aqm-dualq-coupled-08 (Nov 2018) (Work in Progress)
- [RFC7560] Kühlewind, M., Scheffenegger, R. & Briscoe, B. “Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback” IETF RFC7560 (2015)
- [ietf-AccECN] Briscoe, B., Scheffenegger, R. & Kühlewind, M., "More Accurate ECN Feedback in TCP," IETF Internet Draft draft-ietf-tcpm-accurate-ecn-07 (Jul 2018) (Work in Progress)
- [ietf-RACK] Cheng, Y., Cardwell, N., Dukkipati, N. & Jha, P., “RACK: a time-based fast loss detection algorithm for TCP” IETF Internet Draft draft-ietf-tcpm-rack-04 (Jul 2018) (Work in Progress)
- [ietf-ECN++] Bagnulo, M. & Briscoe, B., “ECN++: Adding Explicit Congestion Notification (ECN) to TCP Control “Packets IETF Internet Draft draft-ietf-tcpm-generalized-ecn-03 (Oct 2018) (Work in Progress)

DOCSIS

- [LLDOCSIS-spec] DOCSIS® 3.1 MAC and Upper Layer Protocols Interface (MULPI) Specification (i17+)
- [LLDOCSIS-overview] White, G., Sundaresan, K., and Briscoe, B. “Low Latency DOCSIS: Technology Overview” CableLabs White Paper (Feb 2019)

Research papers

- [DCttH] De Schepper, K., Bondarenko, O., Tsang, I.-J. & Briscoe, B., "`Data Centre to the Home': Deployable Ultra-Low Queuing Delay for All," RITE Project Technical report (June 2015)
- [PI2] De Schepper, K., Bondarenko, O., Tsang, I.-J. & Briscoe, B., "PI^2 : A Linearized AQM for both Classic and Scalable TCP," In: Proc. ACM CoNEXT 2016 pp.105-119 ACM (December 2016)
- [L4S-MMSYS] Bondarenko, O., De Schepper, K., Tsang, I.-J., Briscoe, B., Petlund, A. & Griwodz, C., "Ultra-Low Delay for All: Live Experience, Live Analysis," In: Proc. ACM Multimedia Systems; Demo Session pp.33:1-33:4 ACM (May 2016)
- [DCTCP] Alizadeh, M., Greenberg, A., Maltz, D.A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S. & Sridharan, M., "Data Center TCP (DCTCP)," Proc. ACM SIGCOMM'10, Computer Communication Review 40(4):63--74 (October 2010)
- [DCTCP-analysis] Alizadeh, M., Javanmard, A. & Prabhakar, B., "Analysis of DCTCP: Stability, Convergence, and Fairness," In: Proc. ACM SIGMETRICS'11 (2011)
- [CC-scaling-tensions] Briscoe, B. & De Schepper, K., "Resolving Tensions between Congestion Control Scaling Requirements," Simula Technical Report TR-CS-2016-001 (July 2017)
- [low-RTT-scaling] Briscoe, B. & De Schepper, K., "Scaling TCP's Congestion Window for Small Round Trip Times," BT Technical report TR-TUB8-2015-002 (May 2015)
- [paced-chirping] Misund, Joakim and Briscoe, Bob, “Paced Chirping: Rapid flow start with very low queuing delay” In Proc IEEE Global Internet Symposium 2019 (Apr/May 2019)

Implementing the 'TCP Prague' Requirements for L4S

Q&A

and spare slides

FAQs

Q. Why is IP-ECN=X1 classified into L4S queue?

This will classify CE as L4S,
which could have started as ECT(0) (non-L4S) or ECT(1) (L4S)
but been marked as CE by an upstream AQM

A. Reordering a few packets with lower delay is the safe way round

Codepoint	IP-ECN bits	Meaning
Not-ECT	00	Not ECN-Capable Transport
ECT(0)	10	Classic ECN-Capable Transport
ECT(1)	01	L4S ECN-Capable Transport
CE	11	Congestion Experienced

Q. Isn't the square-root only for TCP Reno, not Cubic?

A. Typical scenarios: Cubic still in Reno-compatibility mode (see next slide)

Q. Isn't an ECN-capable SYN over the Internet a flooding attack vulnerability?

A. No. ECN AQMs disable ECN marking under persistent overload (IETF requirement)

Q. Does fq_CoDel fit into the picture?

A. Yes. As-is, fq_CoDel only gives low latency to low bandwidth flows.
But it could include the last ECN bit in its flow classifier,
and implement a simple immediate AQM in any ECT(1) queue

Q. What about BBR?

A. BBR & L4S have complementary goals, and could be combined into one compound congestion control.

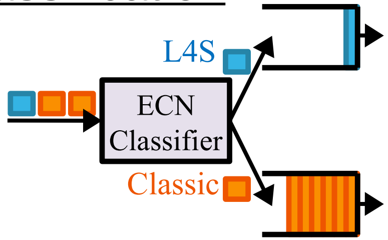
TCP Prague gives much lower delay than BBR, and is much more responsive to other traffic,
but where there is no L4S support in the network TCP Prague is merely no worse than Reno.

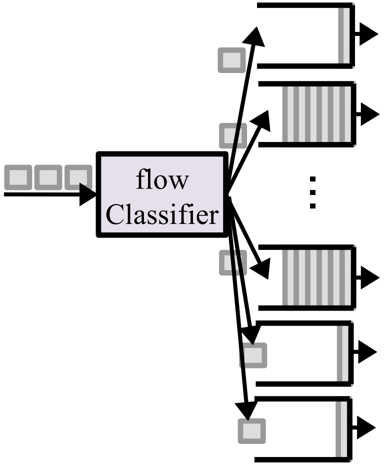
BBR addresses the latter case, where there is no AQM in the network. So if L4S could fall back to BBR
when it detected no L4S support in the network, the user would see lower delay everywhere,
and still get much better performance where there was network support for L4S.

FAQs

- What about the SCE proposal for ECT(1)?
  - Caveat: reverse engineered from unclear initial draft

Classification





Marking

- L4S incremental deployment:
  - ECN used as classifier to isolate low latency flows
  - works with DualQ **or** per-flow Q
  - CE → L4S even if originally ECT(0)

Codepoint	IP-ECN bits	Meaning
Not-ECT	00	Not ECN-Capable Transport
ECT(0)	10	Classic ECN-Capable Transport
ECT(1)	01	L4S ECN-Capable Transport
CE	11	Congestion Experienced

- SCE incremental deployment:
  - ECN not used as classifier
  - need another classifier to isolate low latency flows
  - so only works with per-flow Q

Codepoint	IP-ECN bits	Meaning
Not-ECT	00	Not ECN-Capable Transport
ECT(0)	10	Classic ECN-Capable Transport
SCE	01	Some Congestion Experienced
CE	11	Congestion Experienced

ECN transitions

- RFC3168 & RFC8311
  - ECT(0) → CE
  - ECT(1) → CE
- RFC6040 added support for RFC6660
  - ECT(0) → ECT(1)
- Many encapsulations will still be pre-RFC6040
  - decap will revert ECT(1)
- Ambiguity of CE
  - ECT(0) → CE early on path
    CE → L4S queue later on path
  - 5 unlikely scenarios have to coincide
    to cause an occasional spurious re-xmt

incoming inner	incoming outer			
	Not-ECT	ECT(0)	ECT(1)	CE
Not-ECT	Not-ECT	Not-ECT	Not-ECT	drop Not-ECT
ECT(0)	ECT(0)	ECT(0)	ECT(0)	CE
ECT(1)	ECT(1)	ECT(1)	ECT(1)	CE
CE	CE	CE	CE	CE

Outgoing header (RFC4301 \ RFC3168)

incoming inner	incoming outer			
	Not-ECT	ECT(0)	ECT(1)	CE
Not-ECT	Not-ECT	Not-ECT	Not-ECT	**drop**
ECT(0)	ECT(0)	ECT(0)	**ECT(1)**	CE
ECT(1)	ECT(1)	ECT(1)	ECT(1)	CE
CE	CE	CE	CE	CE

Outgoing header (RFC6040)
(bold = change for all IP in IP)

Distinguishing features of L4S & SCE

	L4S	SCE
No. of queues (see prev slide)	• either dual-queue or per-flow queue	• per-flow queue only
Legacy flow coexistence at:		
• non-ECN bottleneck	• simple to fall back to classic	• simple to fall back to classic
• FQ classic ECN bottleneck	• no prob	• no prob
• 1Q classic ECN bottleneck	• requires heuristic to fall back to classic	• simple to fall back to classic
ECN TCP feedback req'd from recvr	• sndr can force classic ECN rcvr to give fine-grained feedback (unreliable delivery) • or fully works with base AccECN rcvr	• No ECT(1) feedback in TCP without new AccECN TCP option
Additional link benefits	• Enables links to optimize for new sender behavior (e.g. out-of-order tolerance)	
Congestion control status	• DCTCP - widespread deployment experience	• A high-level idea

Low delay *and* high throughput

- ~10x lower delay than state-of the art AQMs
- with capacity-seeking behaviour – TCP-like and adaptive real-time
- for all packets, not just a 'low delay' class

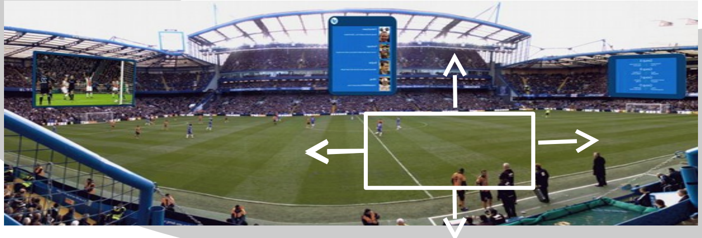
- IETF's experimental track L4S architecture

  Low Latency
  Low Loss
  Scalable throughput
- Also adopted in DOCSIS 3.1

[RFC8311], [ietf-l4s-arch], [LLDOCSIS-spec], [LLDOCSIS-overview]

Ultra-low latency for *every* application

- Not only non-queue-building traffic
  - DNS, gaming, voice, SSH, ACKs, HTTP requests, etc
- Capacity-seeking traffic as well
  - TCP, QUIC, RMCAT for WebRTC
  - web, HD video conferencing, interactive video, cloud-rendered virtual reality, augmented reality, remote presence, remote control, interactive light-field experiences,...

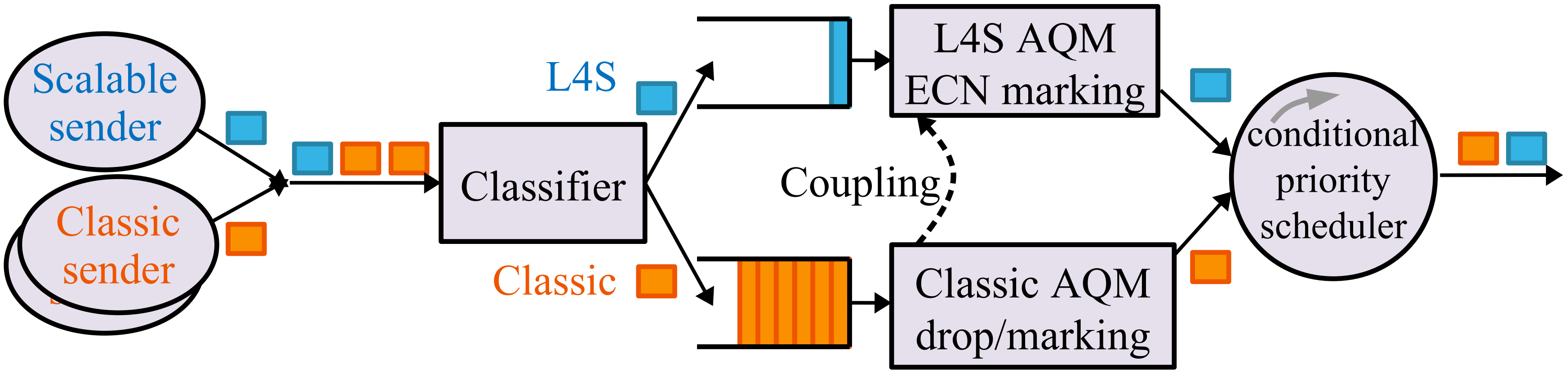


DualQ Coupled AQM

latency isolation, but bandwidth pooling

- L4S-ECN: senders set ECT(1) → classifies into L4S queue

Codepoint	IP-ECN bits	Meaning
Not-ECT	00	Not ECN-Capable Transport
ECT(0)	10	Classic ECN-Capable Transport
ECT(1)	01	L4S ECN-Capable Transport
CE	11	Congestion Experienced



see Olga's later talk: DualPI² qdisc

[RFC8311], [ietf-ecn-l4s-id], [dualpi2-netdev], [ietf-dualq-aqm]

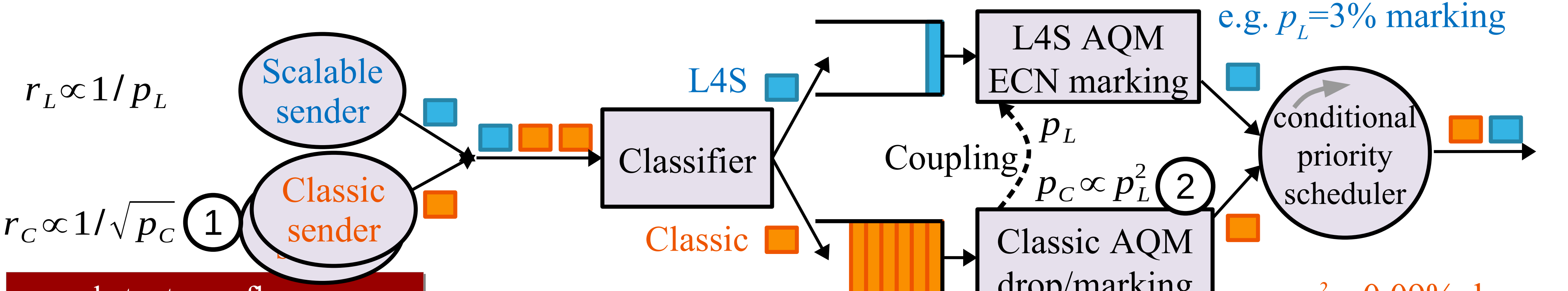
DualQ Coupled AQM

latency isolation, but bandwidth pooling

- flow rate 'fairness' across the two queues

  1 classic congestion control (TCP & QUIC):
  rate depends on the square root of the drop level

  2 counterbalanced by the squaring



r: packet rate per flow
p: drop or marking probability

see Olga's later talk: DualPI² qdisc

- no flow ID inspection, no bandwidth priority

[RFC8311], [ietf-ecn-l4s-id], [dualpi2-netdev], [ietf-dualq-aqm]

Potential Minor change to FQ-CoDel to support L4S

- Don't need to couple AQMs if you have FQ
- FQ-CoDel target used to be lower for ECN-capable flows
- A simple patch could use a shallow target for ECT(1) flows
  - L4S AQM is stateless: just immediate shallow threshold ECN marking
- Open question
  - Whether/how to schedule L4S flows
    with latency priority but not bandwidth priority