Low Latency Low Loss Scalable Throughput (L4S)

draft-ietf-tsvwg-l4s-arch-04
draft-ietf-tsvwg-ecn-l4s-id-07
draft-ietf-tsvwg-aqm-dualq-coupled-10

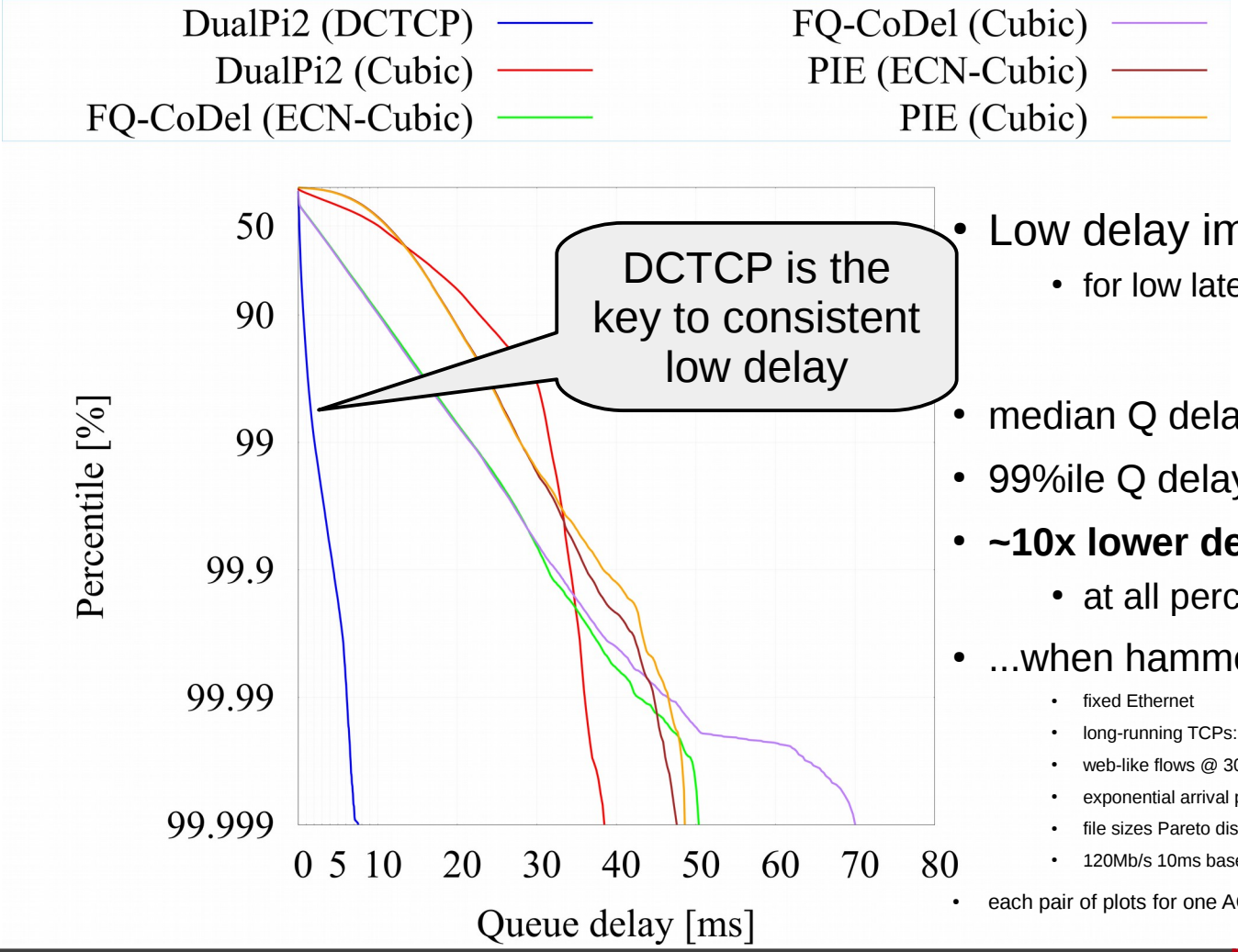
Bob Briscoe, CableLabs <ietf@bobbriscoe.net>
Koen De Schepper, NOKIA Bell Labs <koen.de_schepper@nokia.com>

TSVWG, IETF-105, Jul 2019

Motivation

- Ultra-low queuing delay
  for *all* Internet applications
- including capacity-seeking (TCP-like)

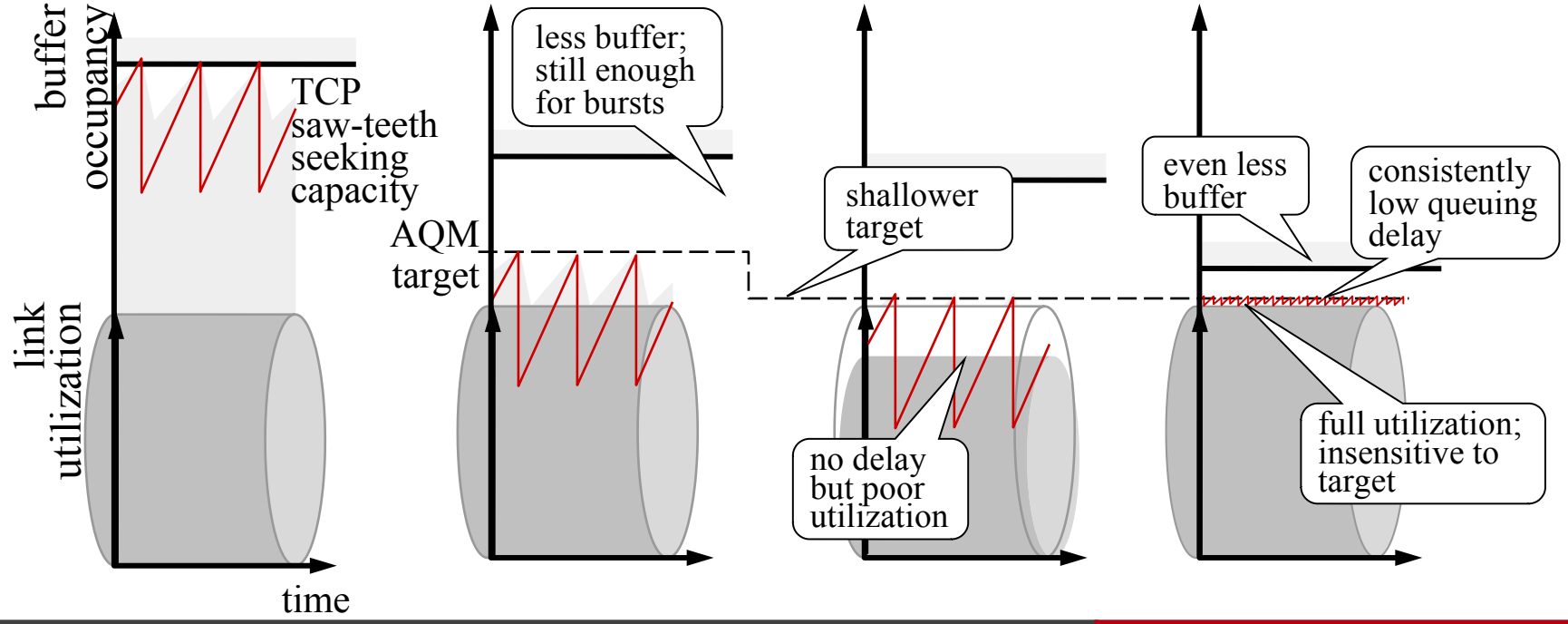
TCP Performance



- Low delay important at higher %-iles
  - for low latency real-time delivery
- median Q delay: 100-200µs
- 99%ile Q delay: 1-2ms
- **~10x lower delay than best 2nd gen. AQM**
  - at all percentiles
- ...when hammering each AQM
  - fixed Ethernet
  - 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
- each pair of plots for one AQM is one experiment run

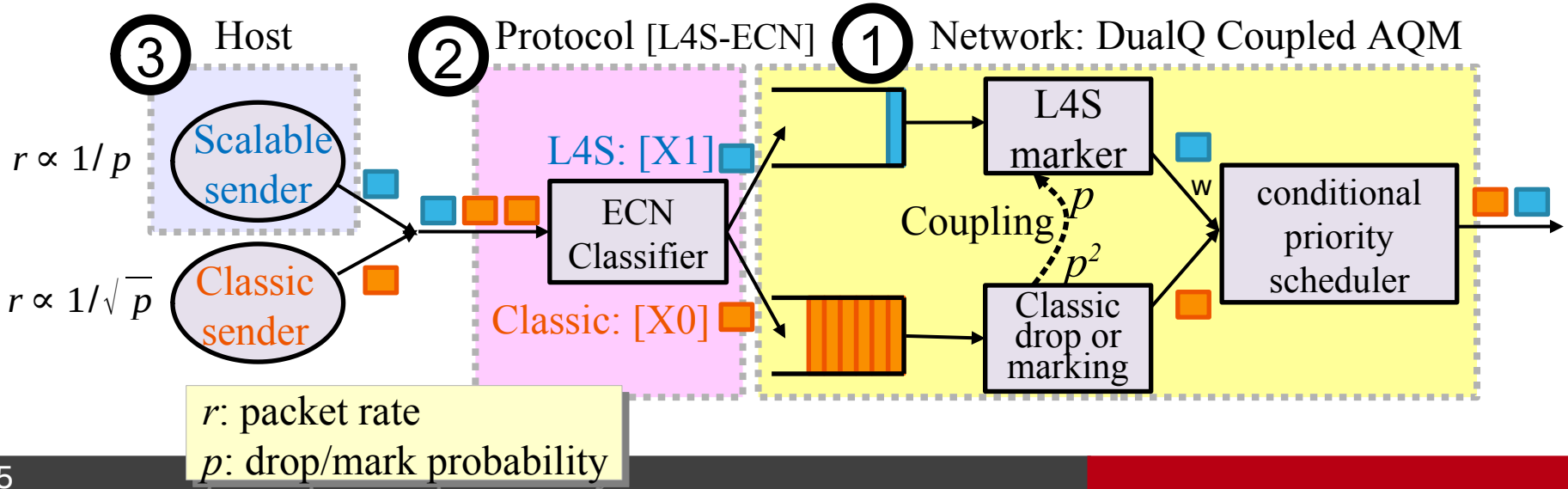
The trick: scalable congestion control

	(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)



Coexistence #1

- Problem
  - Scalable congestion controls more aggressive than 'Classic' (TCP-Friendly)
- Solution **without** flow inspection: Dual Queue Coupled AQM
  - Counter-balance with more aggressive ECN-marking



Coexistence #2

- Solution **with** flow inspection: FQ_xxx_L4S: simple patch
  - If ECT(1), shallow threshold (or ramp) marking
    - based on immediate queue – stateless (no smoothing)
    - else mark/drop with xxx (CoDel, PIE, etc...)
- Description added to draft-ietf-tsvwg-l4s-arch

Implementation status

L4S transport protocols

- TCP Prague – Linux Ref
- rmcat over RTP: L4S-SCReAM
- QUIC Prague
- BBRv2

L4S Transport Components

- Linux: ECT(1), ECN++
- Linux & FreeBSD: AccECN
- Ongoing: Paced Chirping, Sub-Pkt-Wnd

DualQ Coupled AQMs

- Linux Ref Impl'n:
  - DualPI2 resubmitting to mainline
- Low Latency DOCSIS:
  - ns3 Coupled DualQ AQM
  - 2 Cable Modem chipset implementations
  - multiple CMTS implementations
- An Ethernet switch chipset:
  - Curvy RED

L4S FQ_CoDel

- Linux patch

Open Source links

- Dual Queue Coupled AQM
  - Linux: https://github.com/L4STeam/sch_dualpi2_upstream
- L4S Demo/Test GUI
  - Linux: https://github.com/L4STeam/l4sdemo
- TCP Prague (ECT(1), ECN++, AccECN)
  - https://github.com/L4STeam/tcp-prague (Linux)
- QUIC Prague
  - https://github.com/qdeconinck/picoquic/tree/quic-prague (Linux, FreeBSD, Windows)
- SCReAM with L4S support
  - https://github.com/EricssonResearch/scream (Linux, FreeBSD, Windows)
- BBRv2 with L4S support
  - https://github.com/google/bbr/blob/v2alpha/README.md (Linux)
- Paced Chirping (proof-of-concept Linux research code)
  - https://github.com/JoakimMisund/PacedChirping

Hackathon

- L4S Interop testbed (Olivier Tilmans, Koen De Schepper)
- L4S Flent regression tests (Pete Heist, Jonathan Morton, Rodney Grimes)
    - Integration with L4S testbed, validated L4S GUI results
- FreeBSD AccECN implementation (Richard Scheffenegger, Michael Tuexen)
- FreeBSD/Linux AccECN interop (Richard Scheffenegger, Olivier Tilmans)
    - first AccurateECN interop connection on the Internet (IETF network)
- TCP-Prague sub-packet-window (Asad Ahmed)
- TCP-Prague Paced Chirping in ns3 (Tom Henderson, Joakim Misund)

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**	**mandatory**
Accurate ECN TCP feedback	**sysctl option**	**?**	**mandatory**
Reno-friendly on loss		**inherent**	**inherent**
Reno-friendly if classic ECN bottleneck			open issue
Reduce RTT dependence			simulated
Scale down to fractional window	thesis write-up	thesis write-up	thesis write-up
Detecting loss in units of time	**default RACK**	**default RACK**	mandatory?
Optimizations			
ECN-capable TCP control packets	**module option off**	**on**	**default off→on later**
Faster flow start	in progress		
Faster than additive increase		in progress	

Recent developments #1

DualPI2 parameter auto-calc

- for Internet: Zero config – just use defaults
- for uncommon deployments (eg. DC)
  - front-end to auto-calculate 4 parameters

meaningful input parameters	raw input parameters
	target: queue delay
RTT_typ: typical RTT	Tupdate: sampling interval
RTT_max: maximum RTT	alpha: PI integral gain
	beta: PI proportional gain

Recent developments #2

Queue Protection function

draft-briscoe-docsis-q-protection-00

- Informational write-up of DOCSIS algo for the IETF community:
  - pseudocode already published in DOCSIS spec.
  - this adds context & explanation, and
  - **objective definition of flow behaviour necessary to avoid packet rejection**
- Not one of the core L4S drafts – not even a tsvwg draft
  - overload protection [aqm-dualq-coupled Appx A.2] likely a sufficient alternative
  - during the L4S experiment we'll see if it'is necessary (can be disabled in DOCSIS)
- V simple per-flow algo at enqueue to the DOCSIS Low Latency (L4S) queue
  - in normal circumstances, does nothing except monitor
  - maintains per-flow* queuing scores that allocate responsibility for excess queuing
  - the more a configured Q delay threshold is exceeded,
    the more packets from high-scoring flows will be rejected

* flow state expires between a flow's packets, except for ill-behaved flows

Open issues #1: RFC3168 ECN in a FIFO

- Nov 2016, after 16 months of deliberation
  - WG chose ECT(1) for L4S ECN
  - CE ambiguous, but least worst compromise
  - L4S ECN coexists with 3168 ECN, if it's all FQ
- All academic ECN studies over the years (incl. 2017, 2019) found virtually no CE marking
  - using active measurement
- Mar 2017 study by Apple found CE marking
  - using passive measurement

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

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

Open issues #1:
RFC3168 ECN in a FIFO

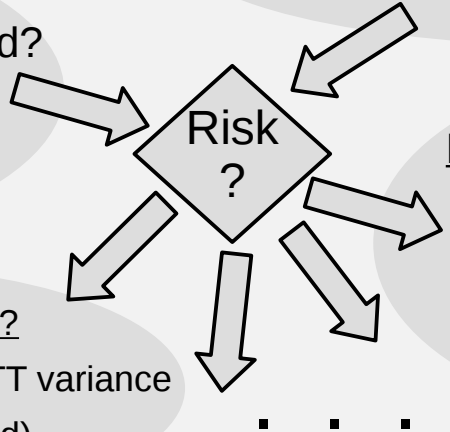
- Assumed all RFC3168 ECN AQMs likely to be FQ_CoDel
  - So L4S traffic would coexist with TCP-Friendly
- What to do if assumption is unsound?

Ground truth

- Any FIFO RFC3168 ECN routers enabled?
  - Two CDNs testing for Echo CE
  - Access to results not assured
- Devised targeted FQ v FIFO test

Quantify flow imbalance

- Testbed measurements (next slide)



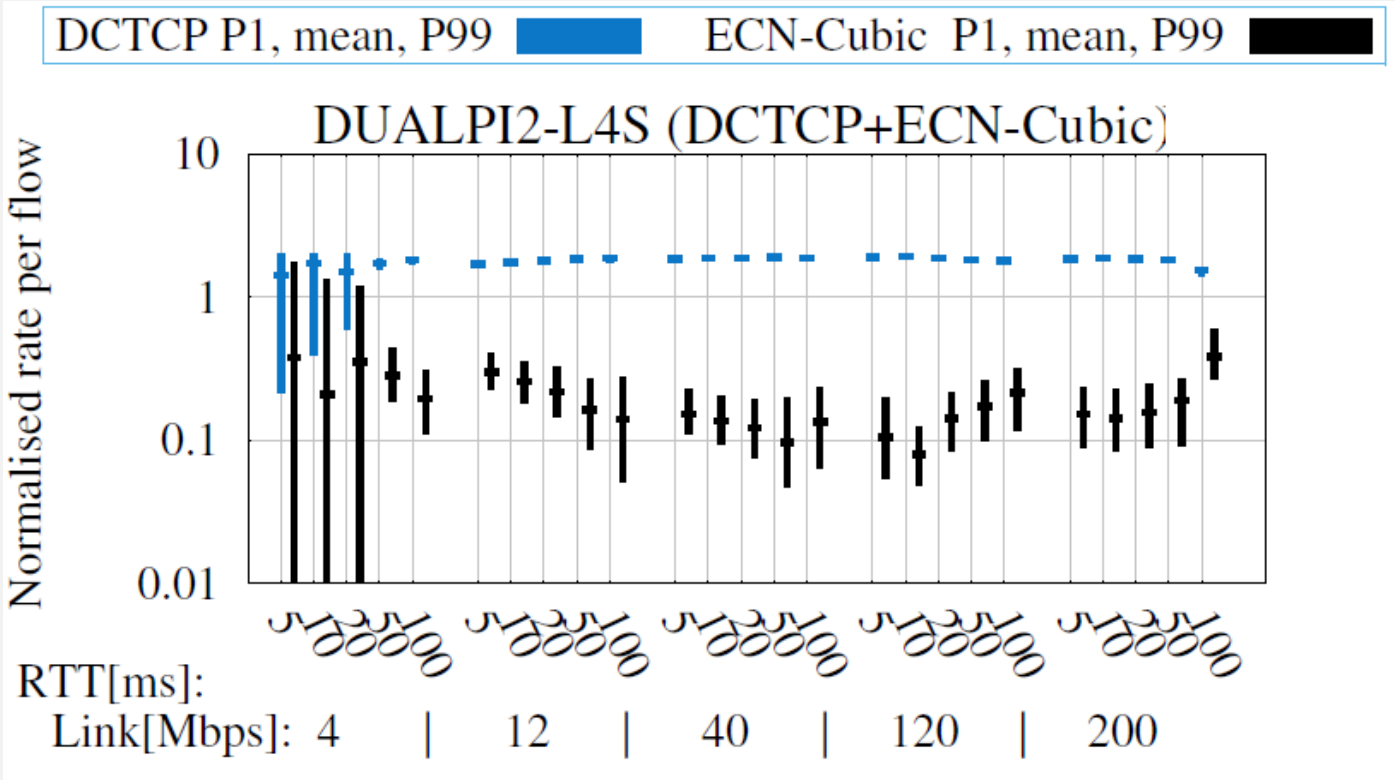
Lo-risk, add advice to L4S expt:

- Limit experiment over your networks (e.g. disable on CDN ports) if RFC3168 AQM is or will be deployed

Hi-risk: Run-Time Detection?

- L4S sender Measures RTT variance
- (To be implemented/tested)

Open issues #1:
RFC3168 ECN in a FIFO



Open issues #2

Loss detection in time units

- Objections and proposed fixes:

1) 'MUST' could be interpreted as a prohibition of 3DupACK in controlled environments where reordering is vanishingly small anyway
   - new wording proposed

2) Overloads one codepoint with two architecturally distinct functions: low queuing delay & low resequencing delay
   - Consider value vs cost of 2 independent identifiers

3) One experiment (L4S) depending on another (RACK)
   - Underlying concern: to avoid L4S success depending on a failed experiment
   - If RACK fails (it's already widely deployed), this aspect of L4S can be relaxed
   - Note: dependency on the *idea* under RACK, not a normative reference

Open issues #2

Loss detection in time units

- Ways forward (for WG to decide):
  - Write as a MUST or a SHOULD?
  - Warn that service could degrade if ignore SHOULD
  -

L4S status update: IETF specs

Deltas since last IETF in Red

tsvwg

- L4S Internet Service: Architecture <draft-ietf-tsvwg-l4s-arch-04> [UPDATE]
- Identifying Modified ECN Semantics for Ultra-Low Queuing Delay (L4S) <draft-ietf-tsvwg-ecn-l4s-id-07> [UPDATE]
- DualQ Coupled AQMs for L4S: : <draft-ietf-tsvwg-aqm-dualq-coupled-10> [UPDATE]
- Interactions of L4S with Diffserv <draft-briscoe-tsvwg-l4s-diffserv-02>
- Identifying and Handling Non-Queue-Building Flows in a bottleneck link draft-white-tsvwg-nqb-02 [UPDATE]
- Low Latency DOCSIS - Technology Overview draft-white-tsvwg-lld-00
- DOCSIS Low Latency Queue Protection draft-briscoe-docsis-q-protection-00 [NEW]
- enabled by <RFC8311> [RFC published]

tcpm

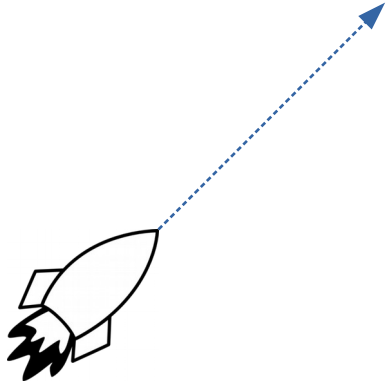
- scalable TCP algorithms, e.g. Data Centre TCP (DCTCP) <RFC8257>, TCP Prague
- Accurate ECN: <draft-ietf-tcpm-accurate-ecn-09> [UPDATE]
- ECN++ Adding ECN to TCP control packets: <draft-ietf-tcpm-generalized-ecn-04> [UPDATE]

Other

- ECN support in trill <draft-ietf-trill-ecn-support-07>, motivated by L4S [RFC Ed Q]
- ECN in QUIC <draft-ietf-quic-transport-22>, [motivated by L4S – Multiple Updates, but not ECN part]
- ECN & Congestion F/b Using the Network Service Header (NSH) <draft-ietf-sfc-nsh-ecn-support-01> [ADOPTED] [supports L4S-ECN]

Next Steps for 3 core L4S drafts

- Classic ECN bottleneck work
- WG Last Call (?)
- Address issues raised



- L4S experiment can start

Low Latency Low Loss Scalable Throughput (L4S)

Q&A

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)