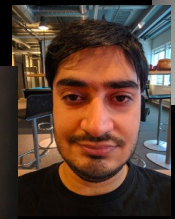
Low Latency Low Loss Scalable Throughput (L4S)

draft-ietf-tsvwg-l4s-arch-06
draft-ietf-tsvwg-ecn-l4s-id-10 {ToDo: 11}
draft-ietf-tsvwg-aqm-dualq-coupled-11

Bob Briscoe, Independent <ietf@bobbriscoe.net>
Koen De Schepper, NOKIA Bell Labs <koen.de_schepper@nokia.com>
Olivier Tilmans, NOKIA Bell Labs <olivier.tilmans@nokia-bell-labs.com>
Greg White, CableLabs <g.white@CableLabs.com>
Asad Sajjad Ahmed, Independent <me@asadsa.com>
Olga Albisser, Simula Research <olga@albisser.org>



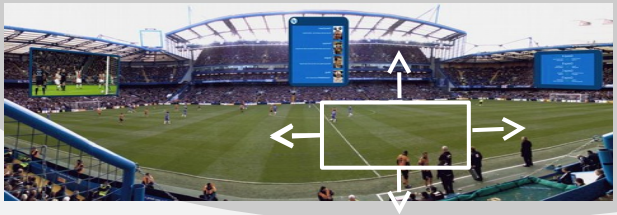


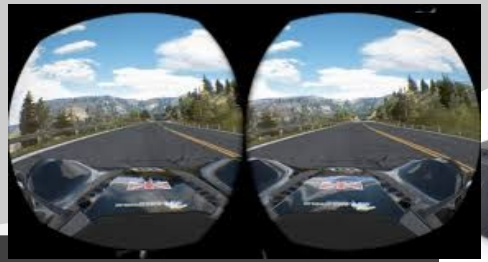



TSVWG, IETF-107, Mar 2020

Ultra-low latency even with high throughput for *all* applications

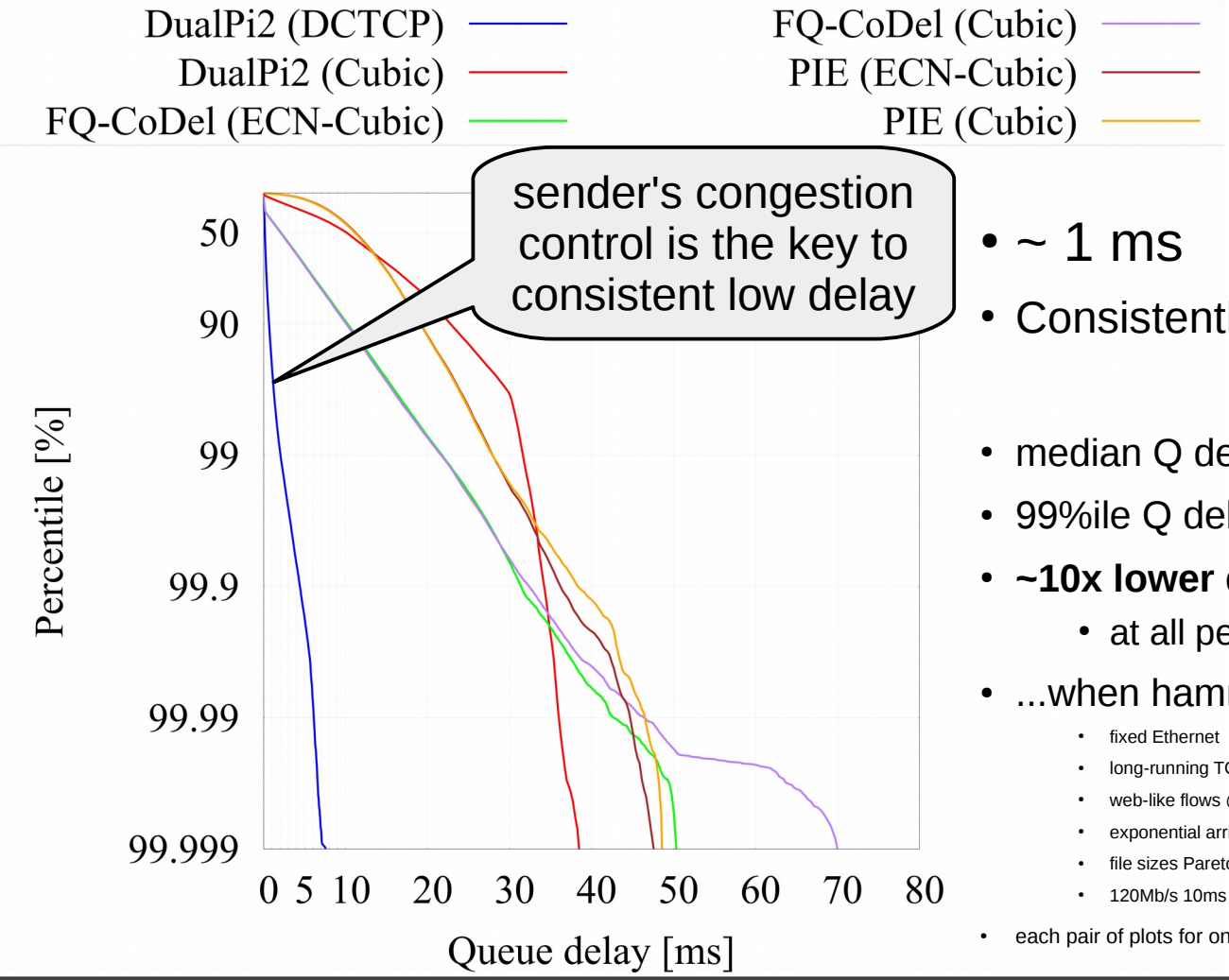
- Not only non-queue-building traffic
  - DNS, gaming, voice, SSH, ACKs, HTTP requests, etc
- Capacity-seeking and adaptive real-time as well
  - TCP, QUIC, RMCAT for WebRTC
  - web, HD video conferencing, interactive video, cloud-rendered virtual reality, augmented reality, remote presence, remote control,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





[L4S-MMSYS]

“Ultra-low” Q delay?

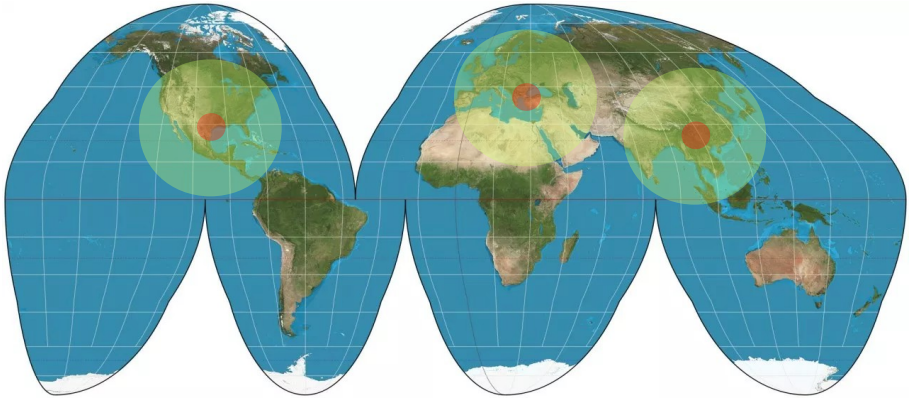


- ~ 1 ms
- Consistently – for real-time apps

- 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

Is such consistently low delay needed?

- Delay budget for 'responsive' feel: 50ms[1]
- L4S: 5x more reach than previous AQMs[2]
  - for each user, or from each data centre
  - Los Angeles to Atlanta, not just to Phoenix



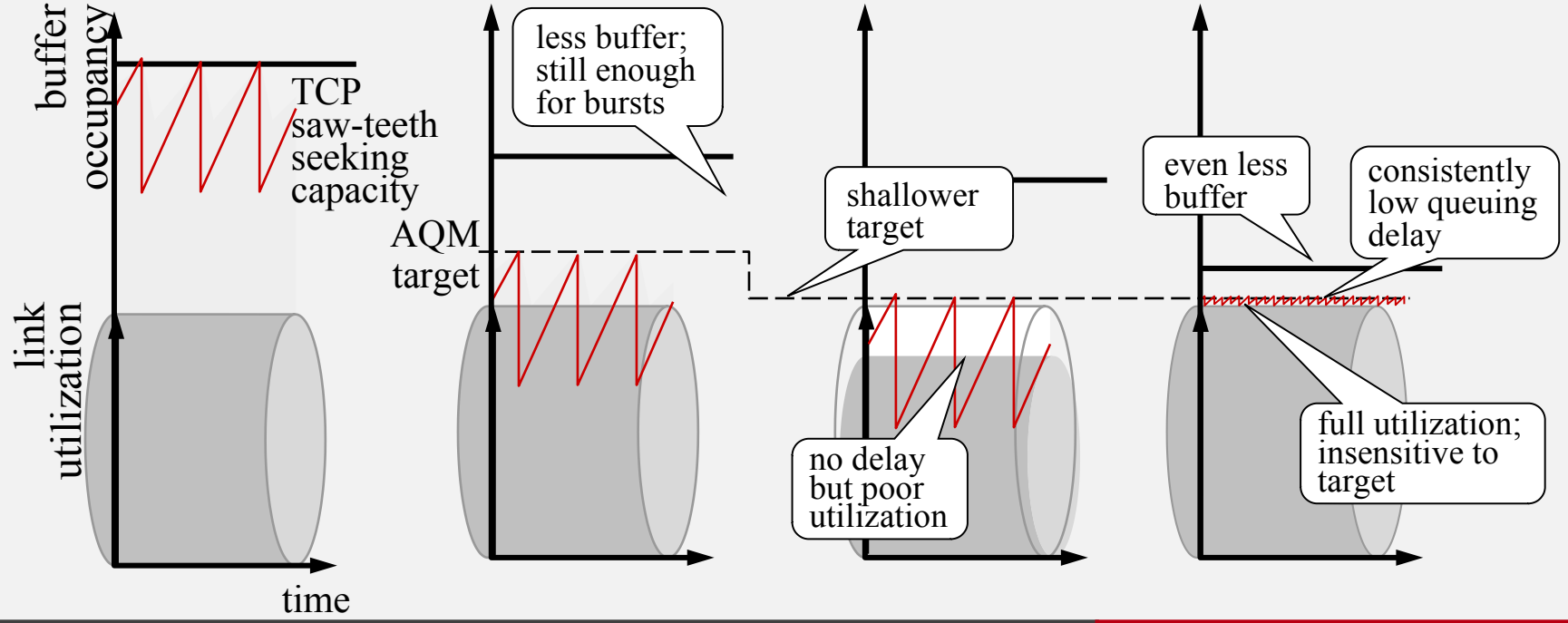
- min non-network delay: 13ms[3]
- leaves (50 - 13 =) 37ms for network

	queue delay (99 %ile)	left for 2-way propagation	reach in fibre	coverage of 'responsive feel' visualized on map
PIE or FQ-CoDel	30ms	37 - 30 = 7ms	700km (440 miles)	
L4S	2ms	37 - 2 = 35ms	3500km (2200 miles)	

1: VR “Oracle” [Carmack13] <20ms latency 'imperceptible', 50ms feels responsive
2: PIE or FQ-CoDel
3: draft-han-iccrg-arvr-transport-problem-01#appendix-A.1.2

The trick: scalable congestion control

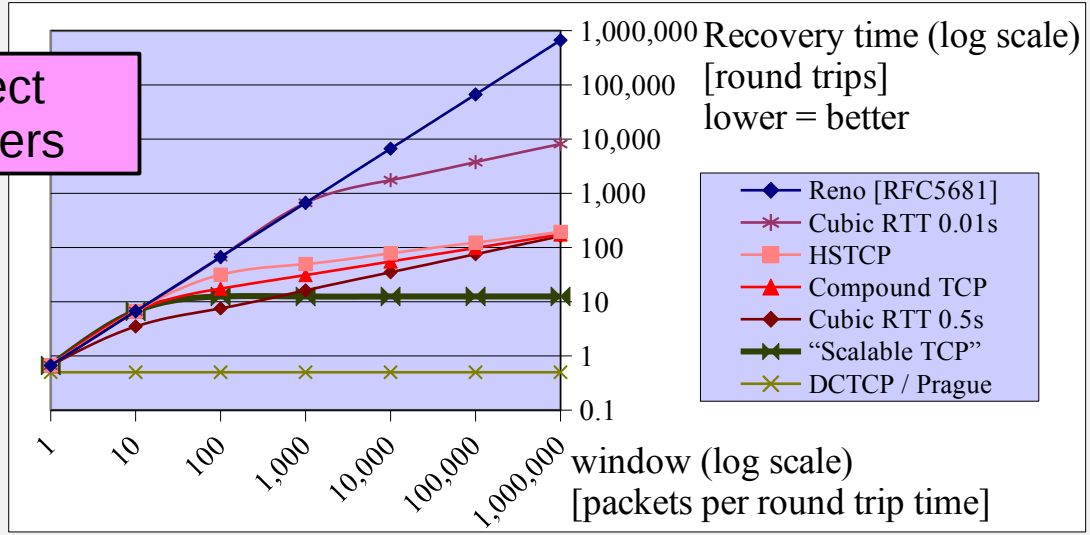
	(1) Today (typical)	(2) Today (at best)	(3) Unacceptable	(4) L4S
Bottleneck	Bloated drop-tail buffer	AQM	Shallower AQM	Immediate AQM
Sender CC	Classic	Classic	Classic	Scalable (tiny saw-teeth)

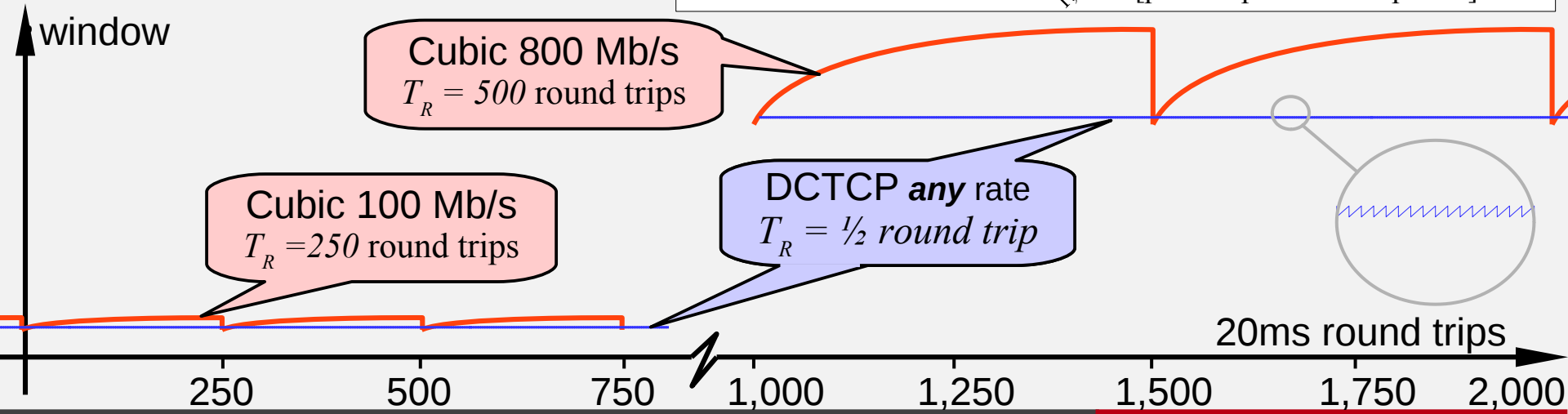


Scalable?

correct numbers

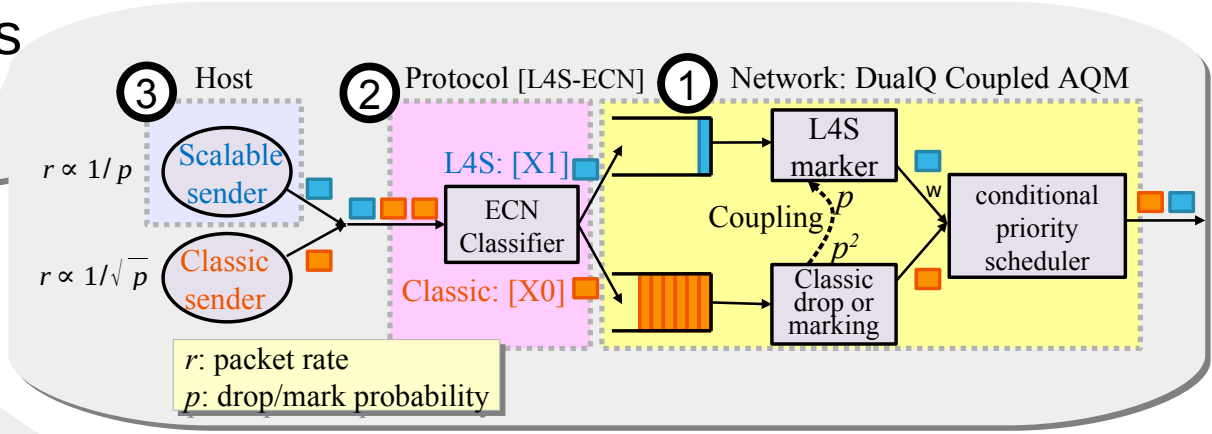
- Duration of sawteeth (recovery time) is invariant as flow rate scales [RFC3649]
  - otherwise problems return in a few years:
    - more queue delay or underutilization
    - more sensitive to disturbance
    - more sluggish at tracking dynamics





Coexistence

- Problem
  - Scalable congestion controls more aggressive than 'Classic' (Reno-Friendly)
- Transition mechanisms
  1) network:
     - dualQ coupled AQM
     - or per-flow queuing
  2) packet identifier:
     - ECT(1) in IP/ECN field
  3) host:
     - Fall-back behaviours on loss or classic ECN



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

Goals: L4S & SCE compared

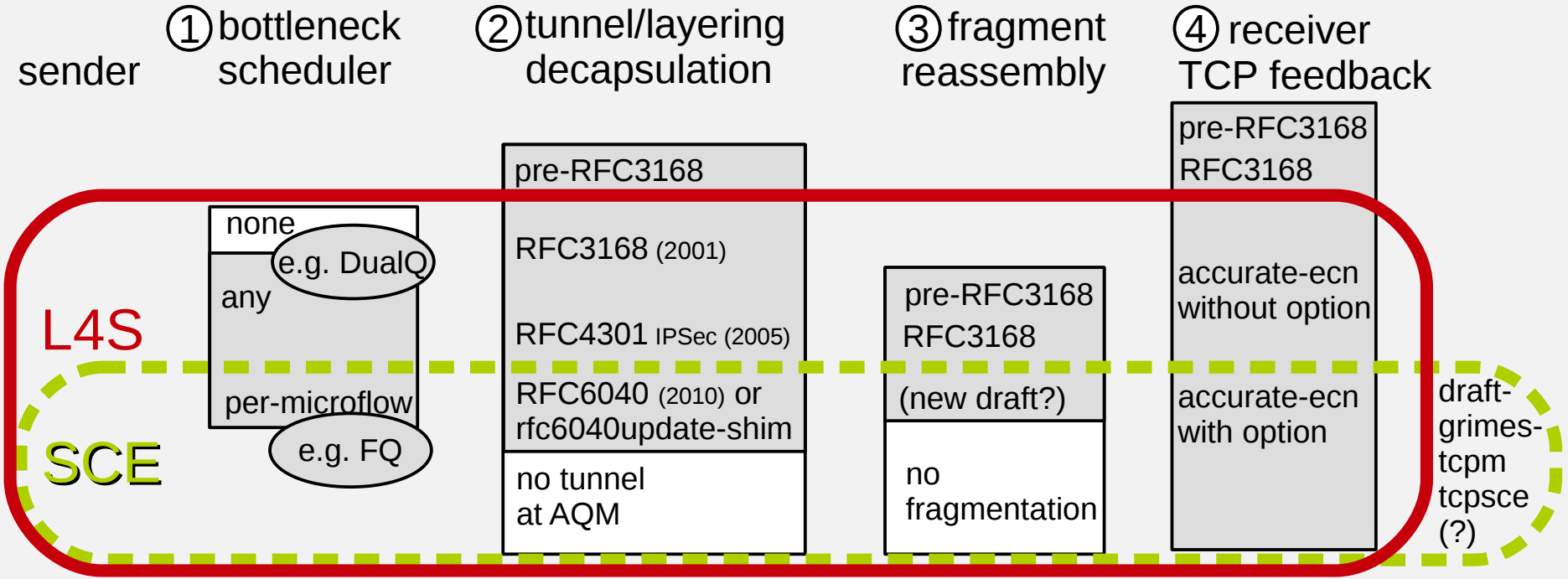
- very low delay with high throughput
- improve utilization of AQMs like FQ-CoDel
- by reducing amplitude of sawteeth

adoption: L4S & SCE compared

- prospect of immediate significant performance improvement
- encourages adoption by operators and their suppliers
- which encourages adoption by the main host OS developers and CDNs
-
-
-
-

- intially build on FQ-CoDel deployment
- introduce CCs that respond to SCE markings; some more scalable than others
- OK if SCE often silently does nothing at first due to tunnels, layering, lack of SCE-TCP receiver, etc
- as sawteeth gradually shrink, FQ AQMs can reduce SCE threshold, as underutilization dilemma decays out
- no solution yet for single queue

Performance Improvement Dependencies: L4S & SCE compared



L4S works with the Internet as it is; SCE needs a new Internet to work

Safety: L4S & SCE compared

- SCE does not have a monopoly on safety

Managing the Experiment:
L4S & SCE compared

- L4S traffic is identifiable
- L4S sender and receiver positively negotiate

- Hard to monitor the experiment
  - only the sender knows which flows are SCE
  - SCE AQM alters some ECT0 to ECT1 whether sender is participating or not
  - compliant receiver feeds back ECT0 / ECT1 whether sender is participating or not
- SCE sender with no SCE feedback could be due to:
  - no SCE logic at bottleneck buffer yet
  - no SCE tunnel decap logic yet
  - no SCE receiver reassembly logic yet
  - no SCE receiver feedback logic yet
  - TCP NS bit not traversing middlebox

What do you lose? L4S & SCE compared

Low Latency DOCSIS (LLD) Status

- Oct'17: LLD Working Group formed
  - 89 individuals, representing:
    Arris/Commscope, Broadcom, CableLabs, Casa Systems, Cisco Systems, Comcast, Cox Communications, Intel, Liberty Global, Nokia, Videotron, Virgin Media, 12 others
  - Integrated L4S DualQ Coupled AQM into DOCSIS3.1 MAC
  - Developed Non-Queue-Building traffic concept and Queue Protection algorithm
  - All features implementable via firmware update to existing DOCSIS 3.1 gear
- Jan'19: LLD Specs published

Interop Events	No. of implementations	
	cable modem (CM) silicon	CM termination system
Sep'19	2	1
Nov'19	2	2
...Three more events planned		

Nokia WiFi Beacon 1 will support L4S

- Full L4S support in Q1-2020
- Other devices planned later
- Using DualPI2 technology
- Demonstrated in BBWF 2019

- Network operators and service providers interested in trails please contact Nokia Digital Home Business Unit

https://www.nokia.com/wifi/beacon-1/

System Architecture WG2

- Proposal to include L4S support
  - Ericsson, Sprint, Google, Nokia Networks, AT&T, Vodafone
- 5G System (5GS) Enhancements for Advanced Interactive Services
  - part of Rel-17 work item 5G_AIS
- AIS, e.g. Cloud Gaming services and Unmanned Aerial Systems,
  - includes requirement for both low latency and high throughput

Open Source

- Dual Queue Coupled AQM
  - Linux: https://github.com/L4STeam/sch_dualpi2_upstream
- L4S Demo/Test GUI
  - Linux: https://github.com/L4STeam/l4sdemo
- network simulator L4S evaluation programmes and scripts
  - ns-3: l4s-evaluation
- TCP Prague
  - https://github.com/L4STeam/tcp-prague (Linux)
- SCReAM (rmcat) 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)
- QUIC Prague
  - https://github.com/qdeconinck/picoquic/tree/quic-prague (Linux, FreeBSD, Windows)
- Paced Chirping (proof-of-concept Linux research code)
  - https://github.com/JoakimMisund/PacedChirping

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	

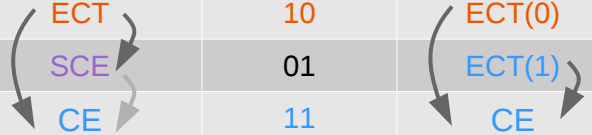
Status against Prague L4S requirements (Nov'19)

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			evaluat'n in progress
Reduce RTT dependence			research code
Scale down to fractional window	research code	research code	research code
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	

SCE vs L4S

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



- L4S primary issue (#16)
  - scalable flows would dominate classic in a Classic ECN AQM without FQ
    - if such things are deployed and if classic fall-back doesn't work
- SCE primary issue
  - per-microflow classification (e.g. FQ) is a necessity
    - otherwise classic dominates SCE

L4S status update: IETF specs

Deltas since last IETF in Red

tsvwg

- L4S Internet Service: Architecture <draft-ietf-tsvwg-l4s-arch-04>
- Identifying Modified ECN Semantics for Ultra-Low Queuing Delay (L4S) <draft-ietf-tsvwg-ecn-l4s-id-08> [UPDATE]
- DualQ Coupled AQMs for L4S: : <draft-ietf-tsvwg-aqm-dualq-coupled-10>
- Interactions of L4S with Diffserv <draft-briscoe-tsvwg-l4s-diffserv-02>
- Identifying and Handling Non-Queue-Building Flows in a bottleneck link draft-ietf-tsvwg-nqb-00 [ADOPTED]
- 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>
- ECN++ Adding ECN to TCP control packets: <draft-ietf-tcpm-generalized-ecn-05> [UPDATE]

Independent Stream

- DOCSIS Low Latency Queue Protection draft-briscoe-docsis-q-protection-00
- Low Latency DOCSIS - Technology Overview draft-white-tsvwg-lld-00

Other

- ECN support in trill <draft-ietf-trill-ecn-support-07>, motivated by L4S [RFC Ed Q]
- ECN in QUIC <draft-ietf-quic-transport-24>, [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> [supports L4S-ECN]

Low Latency Low Loss Scalable Throughput (L4S) Issues

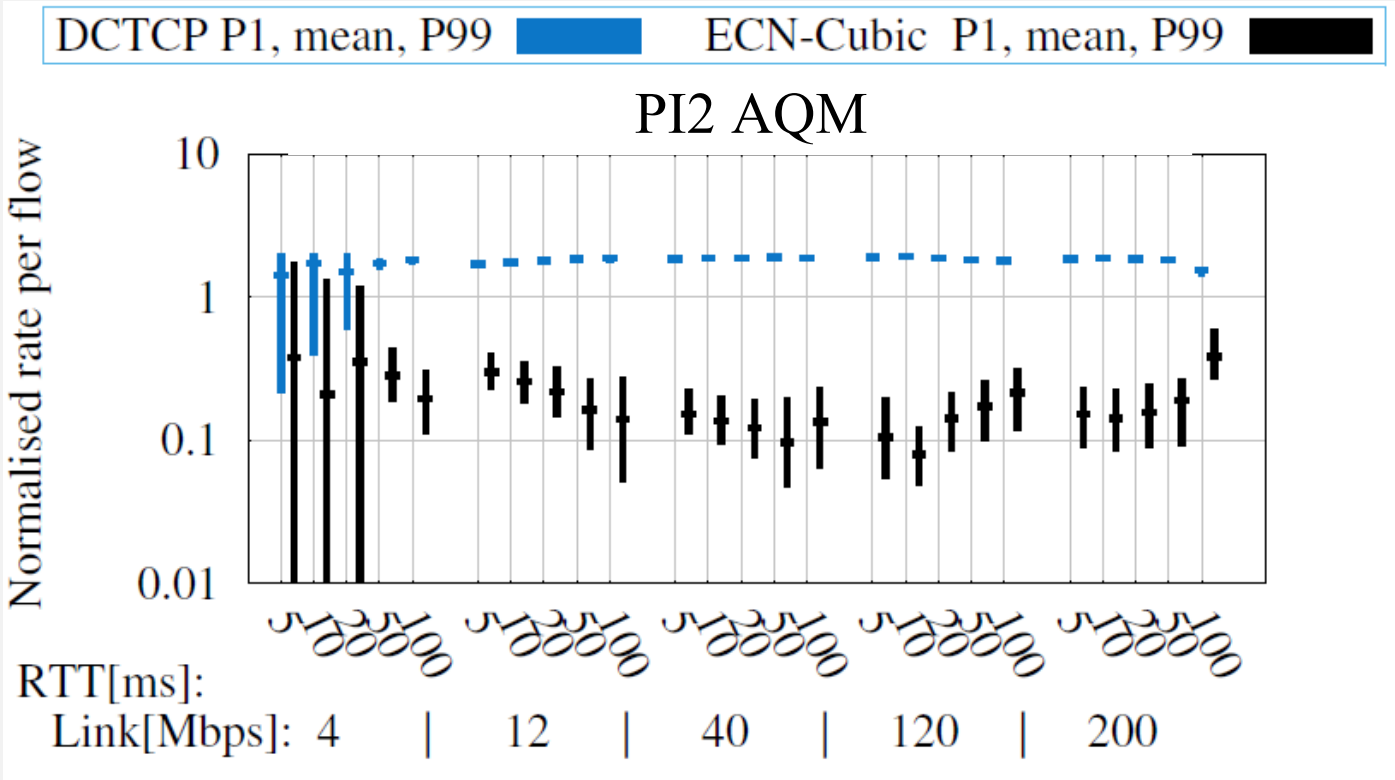
draft-ietf-tsvwg-l4s-arch-04
draft-ietf-tsvwg-ecn-l4s-id-07
draft-ietf-tsvwg-aqm-dualq-coupled-10

Bob Briscoe, Independent <ietf@bobbriscoe.net>
Koen De Schepper, NOKIA Bell Labs <koen.de_schepper@nokia.com>
Olivier Tilmans, NOKIA Bell Labs <olivier.tilmans@nokia-bell-labs.com>
Greg White, CableLabs <g.white@CableLabs.com>



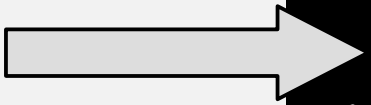

TSVWG, IETF-106, Nov 2019

Issue #16:
RFC3168 ECN AQM in a single Q



Open issues #16: RFC3168 ECN in a single Q

- 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
- Apple identified Argentinian ISP
- Aug 2019 CableLabs contacted ISP
  - ISP not aware of having deployed ECN
  - discovered on most (all?) systems upstream misconfigured to overwrite ToS byte with 0x17
  - Being fixed
- Apple now attempting to identify ISPs



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

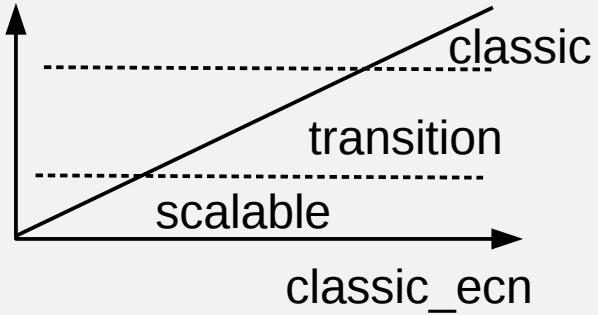
Country	Percentage
United States	0.2
China	1
Mexico	3.2
France	6
Argentine Republic	30

- Marking was mainly seen on the uplink

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

Issue #16:
Fall-back to Reno-Friendly on Classic ECN bottleneck

- Not necessary for ever
  - until RFC3168 ECN superseded (or L4S experiment ends)
- Published Design as a Discussion Paper
  - TCP Prague Fall-back on Detection of a Classic ECN AQM
- Rationale for metrics, pseudocode & analysis
- Detection algorithms – drive a classic ECN AQM score
  - Passive detection algorithm – primarily based on delay variation
  - Active detection technique (if passive raises suspicion)
  - Technique to filter out route-changes (prob. unnecessary)
- Gradual behaviour change-over from scalable to classic
  - e.g. TCP Prague becomes Reno
  - detection unlikely to be perfect

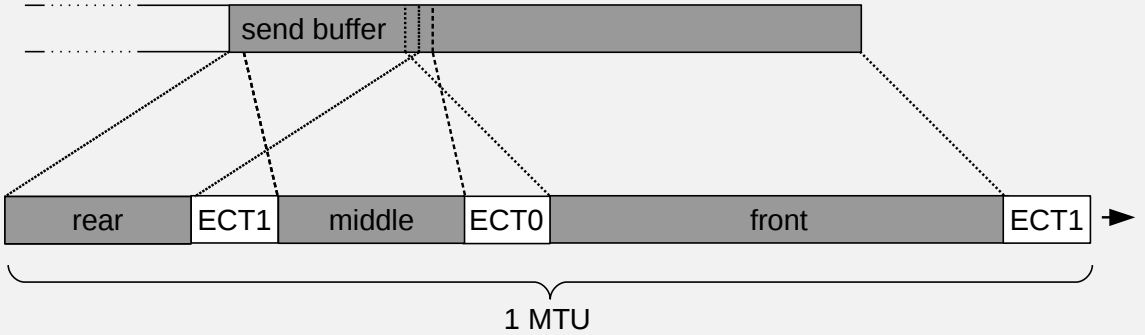


Issue #16:
Fall-back to Reno-Friendly on Classic ECN bottleneck

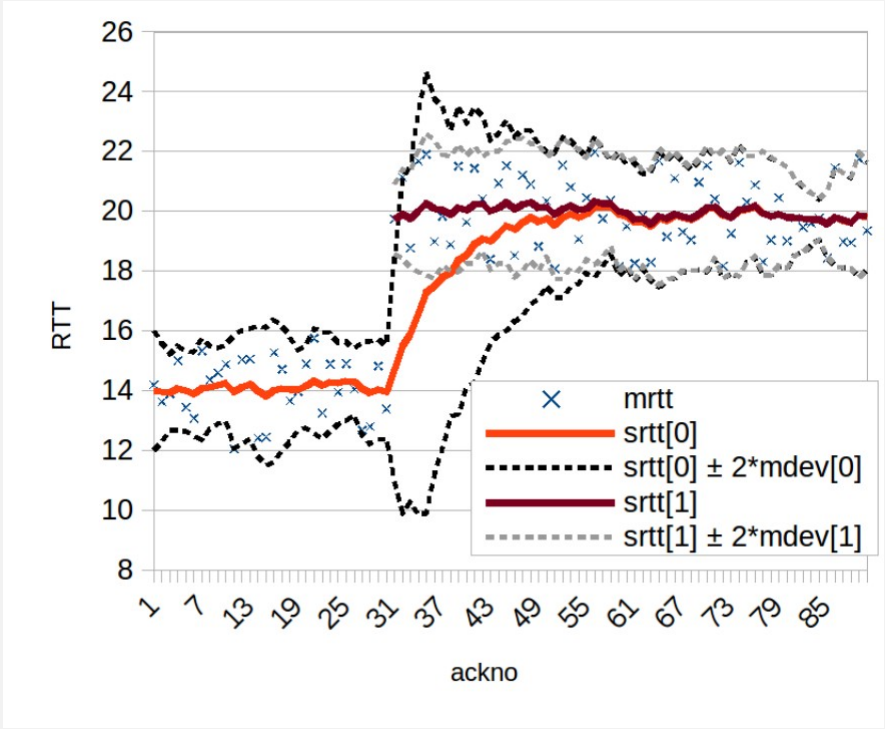
- Passive detection algorithm
  - delayed start following first CE mark
  - 3 weighted elements to detect classic queue
    - mean deviation of the RTT (mdev in TCP)
    - mean Q depth (solely positive factor – min RTT unreliable)
    - degree of self-limiting (app-limited, rwnd-limited) (solely negative factor)
- Implemented
- Evaluation will follow testbed rebuild
  - verifying testbed documentation is sufficient for a newbie

Issue #16:
Fall-back to Reno-Friendly on Classic ECN bottleneck

- Active detection technique
  - if passive raises suspicion,
    - send three overlapping sub-MSS tracer packets
    - forces quick-ACKs
  - if last two reordered, likely L4S
    - reduce suspicion, and continue



Route-change filtering



- if outlier
  - create alt mdev
- unlikely to be necessary
  - mis-measurement too brief to affect passive detection algo

L4S Issue #17 – Interaction w/ FQ AQMs

Here is a simple experiment that should verify the existence and extent of the problem:

Control:

[Sender] -> [Baseline RTT 10ms] -> [FQ_Codel, 100% BW] -> [Receiver]

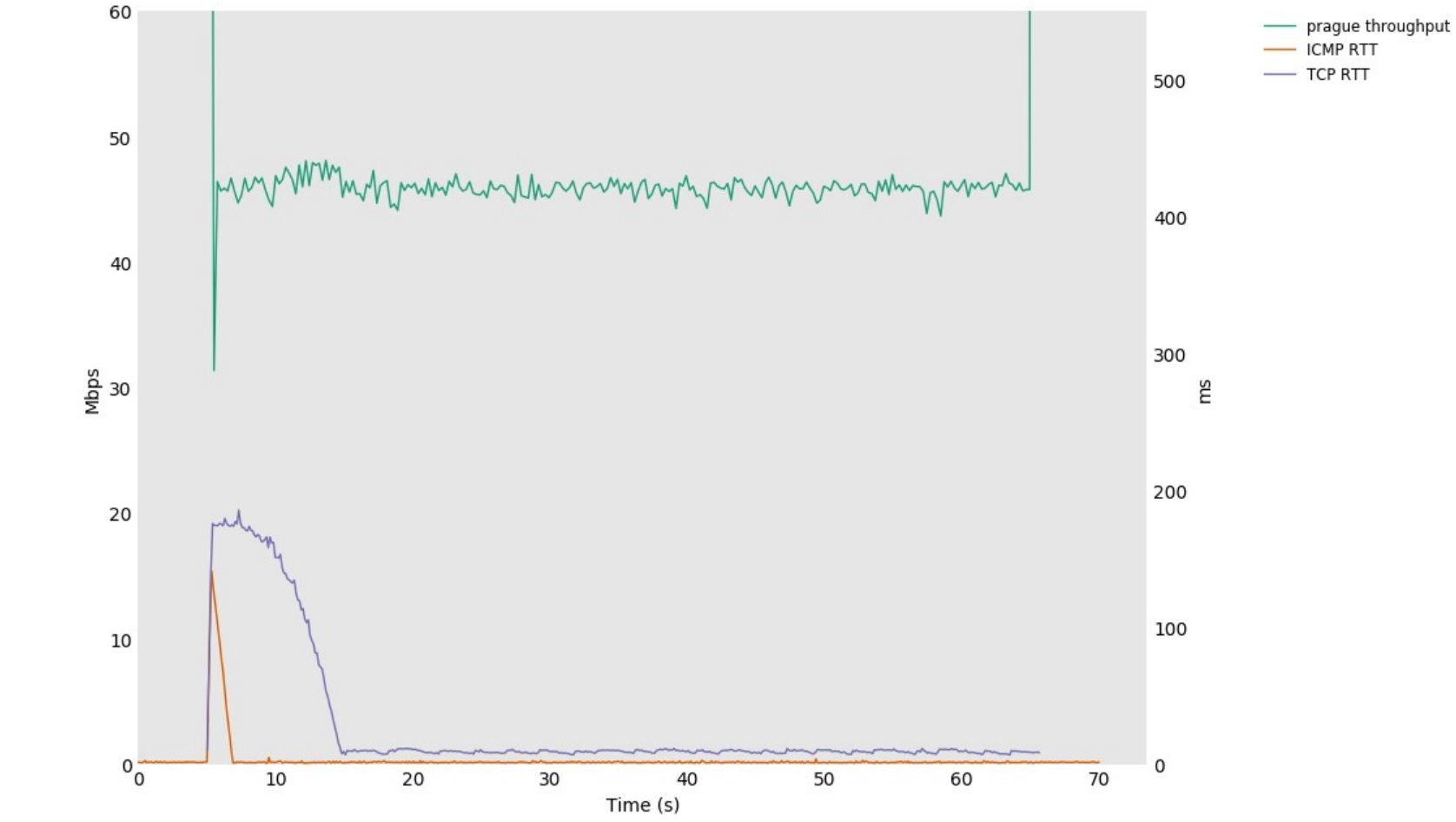
Subject:

[Sender] -> [Baseline RTT 10ms] -> [Dumb FIFO, 105% BW] -> [FQ_Codel, 100% BW] -> [Receiver]

Background traffic is a sparse latency-measuring flow, essentially a surrogate for gaming or VoIP. The instantaneous latency experienced by this flow over time is the primary measurement.

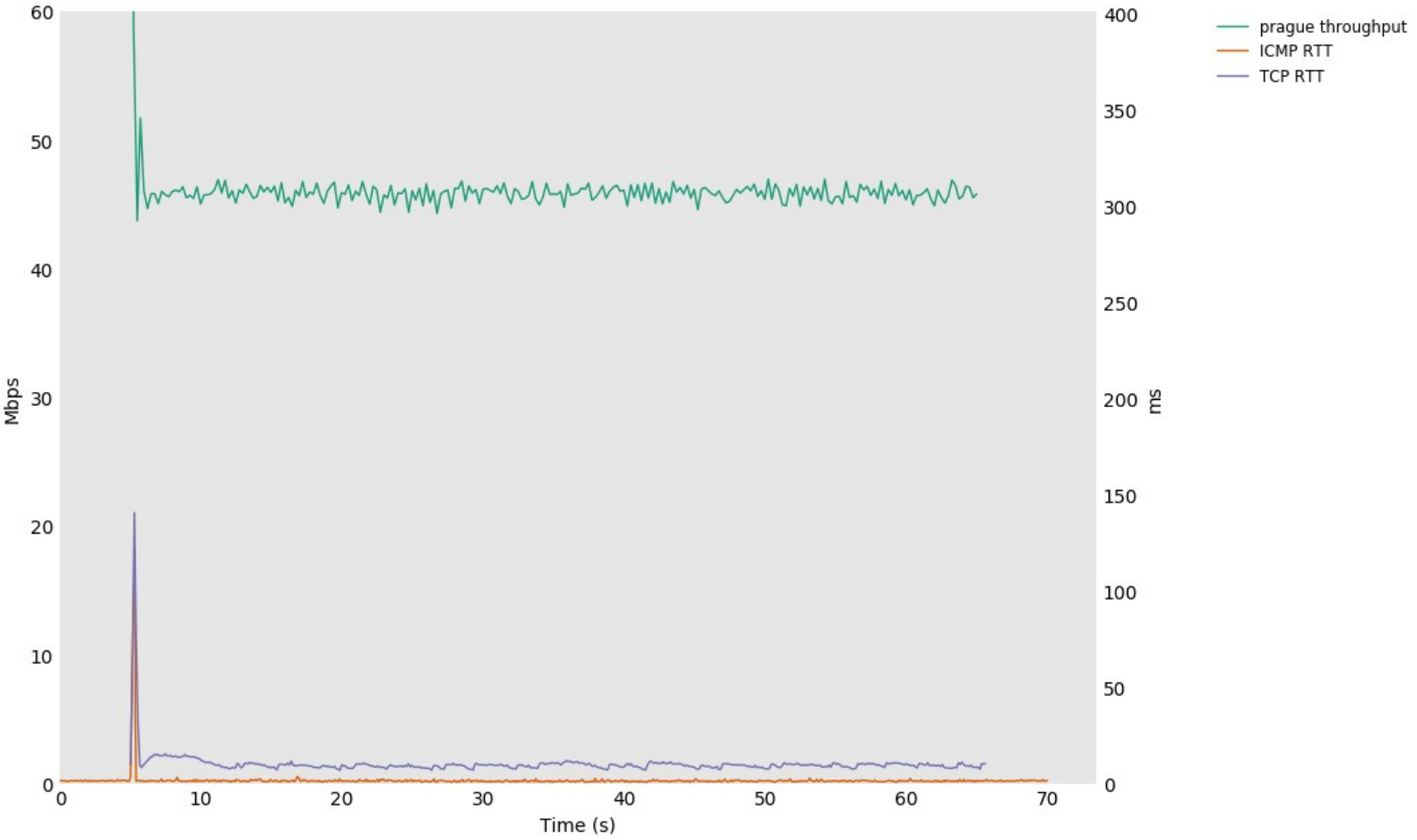
The experiment is simply to start up one L4S flow in parallel with the sparse flow, and let it run in saturation for say 60 seconds. Repeat with an RFC-3168 flow (NewReno, CUBIC, doesn't matter which) for a further experimental control. Flent offers a convenient method of doing this.

Correct behaviour would show a brief latency peak caused by the interaction of slow-start with the FIFO in the subject topology, or no peak at all for the control topology; you should see this for whichever RFC-3168 flow is chosen as the control. **Expected results with L4S in the subject topology, however, are a peak extending about 4 seconds before returning to baseline.**


L4S flent tests (Scenario 6)
Sender → Delay → FIFO (btlneck 1) → FQ-CoDel (btlneck 2) → L4S receiver (initial alpha=0)
b:l4s-s6-prague_alpha cc:prague q:htb(52.5Mbit)+pfifo→htb(50Mbit)+fq_codel bw:50Mbit rtt:0ms
prague throughput
ICMP RTT
TCP RTT
Mbps
ms
Time (s)
0
10
20
30
40
50
60
70
100
200
300
400
500

L4S Issue #17 – Observations

- This result is repeatable using the initial version of TCP Prague, but is not present in DCTCP.
- Discovery:
  - A bug was introduced in the TCP Prague code on July 30 in the initial restructuring to handle pacing/TSO sizing.
  - Prague “alpha” (average fraction of CE bytes) got initialized to zero instead of one.
- All prior testing had been done using DCTCP.
- Upon fixing this bug, the results again match DCTCP

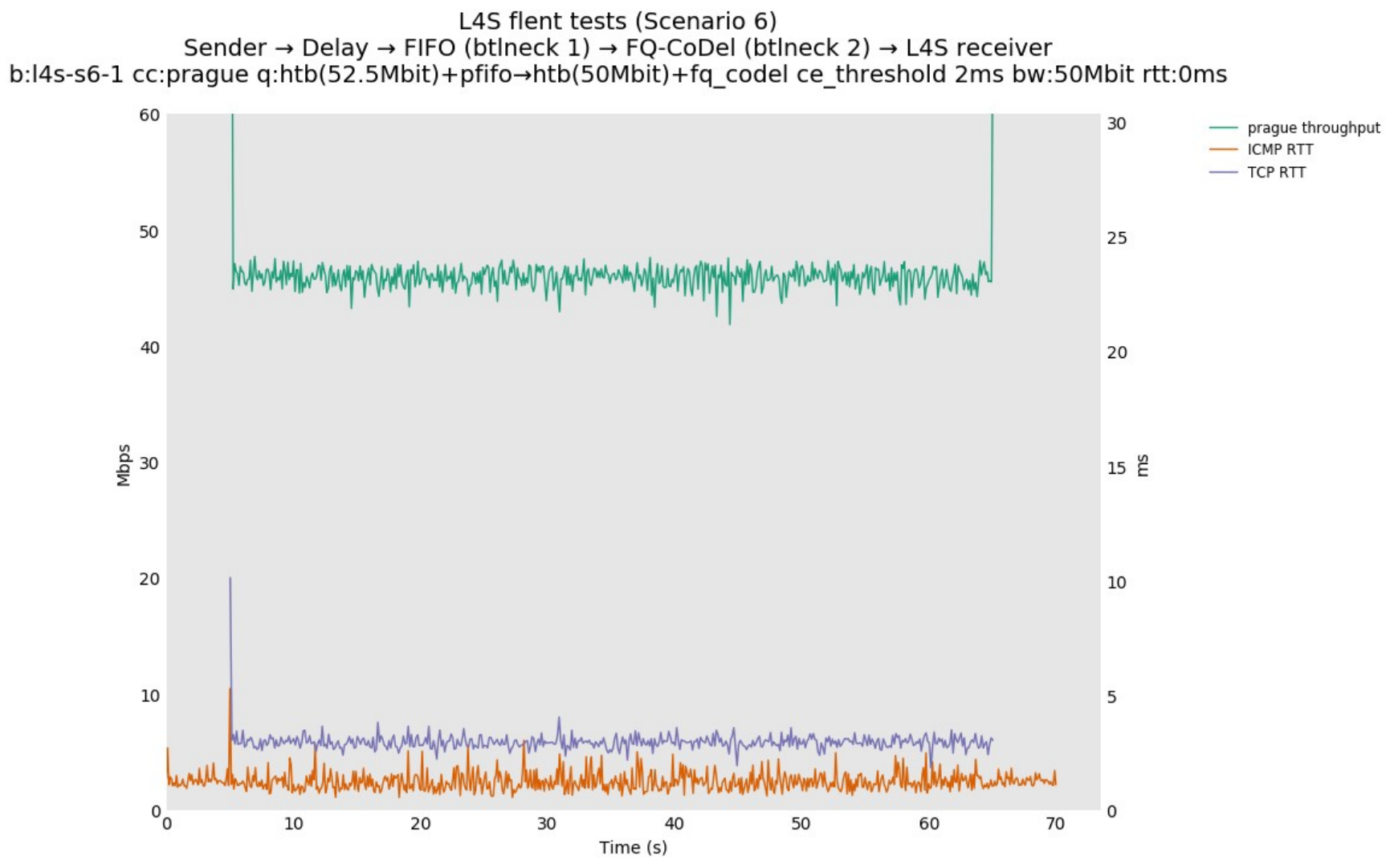

L4S flent tests (Scenario 6)
Sender → Delay → FIFO (btlneck 1) → FQ-CoDel (btlneck 2) → L4S receiver
b:l4s-s6-1 cc:prague q:htb(52.5Mbit)+pfifo→htb(50Mbit)+fq_codel bw:50Mbit rtt:0ms
prague throughput
ICMP RTT
TCP RTT
Mbps
ms
Time (s)
0
10
20
30
40
50
60
70
0
50
100
150
200
250
300
350
400

Issue #17: Conclusion

- The main result of concern was due to a bug in initializing the value of TCP Prague alpha, which has been fixed and demonstrated to resolve the latency impact that was spanning multiple seconds

- The remaining short duration latency spike in the FIFO queue is seen in all congestion control variants tested, including BBRv1, NewReno, and Cubic, and is not specific to Prague

- If the CoDel queue is upgraded to perform Immediate AQM on L4S flows, the latency spike can be largely avoided.


L4S flent tests (Scenario 6)
Sender → Delay → FIFO (btlneck 1) → FQ-CoDel (btlneck 2) → L4S receiver
b:l4s-s6-1 cc:prague q:htb(52.5Mbit)+pfifo→htb(50Mbit)+fq_codel ce_threshold 2ms bw:50Mbit rtt:0ms
prague throughput
ICMP RTT
TCP RTT
Mbps
ms
Time (s)
0
10
20
30
40
50
60
70
5
15
25

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

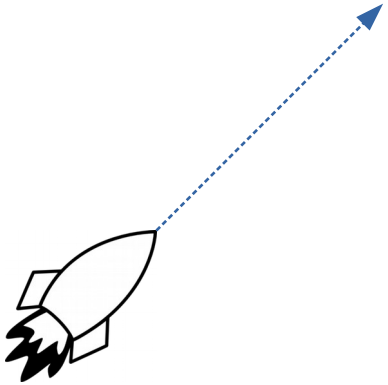
Open issues #2

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
  -

Next Steps for 3 core L4S drafts

- Update architecture to improve language
- Editorial updates to all 3
    - esp. time-units issue
- WGLC



Low Latency Low Loss Scalable Throughput (L4S)

Q&A

spare slides

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)

performance improvement dependencies: L4S & SCE compared