Low Latency Low Loss Scalable Throughput (L4S)

TCP Prague Status pt2
draft-ietf-tsvwg-ecn-l4s-id

Bob Briscoe, Independent <ietf@bobbriscoe.net>
about the work of people too numerous to list

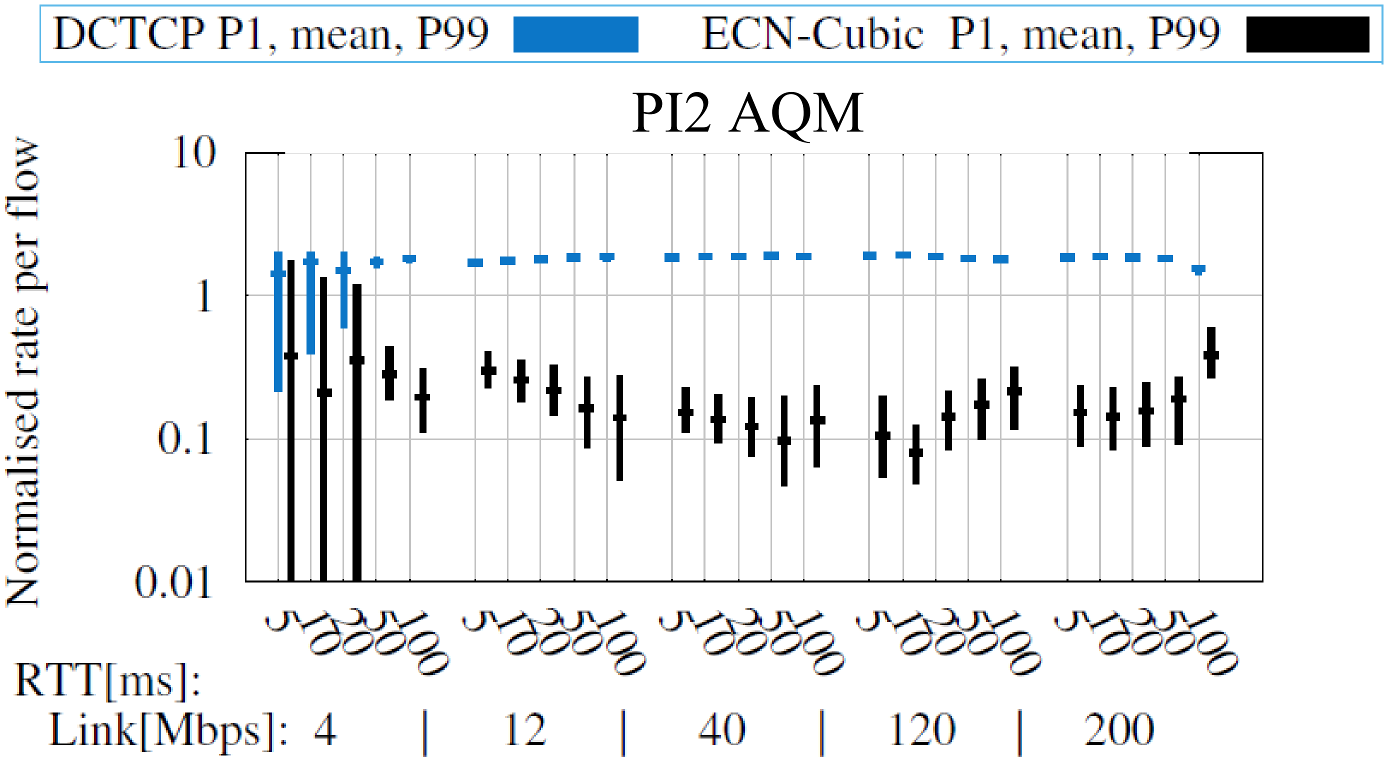
TSVWG, IETF-106, Nov 2019

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 draft IETF conditions for setting ECT(1) in IP
  - draft-ietf-tsvwg-ecn-l4s-id

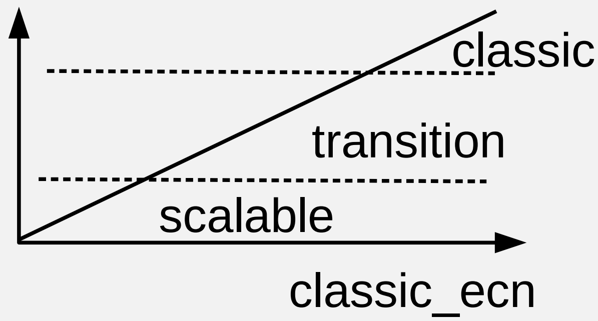
Requirements	
	L4S-ECN Packet Identification: ECT(1)
	Accurate ECN TCP feedback
	Reno-friendly on loss
	Reno-friendly if Classic ECN bottleneck
	Reduce RTT dependence
	Scale down to fractional window
	Detecting loss in units of time
Optimizations	
	ECN-capable TCP control packets
	Faster flow start
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

Issue #16: RFC3168 ECN AQM in a single Q



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  - until RFC3168 ECN superseded (or L4S experiment ends)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

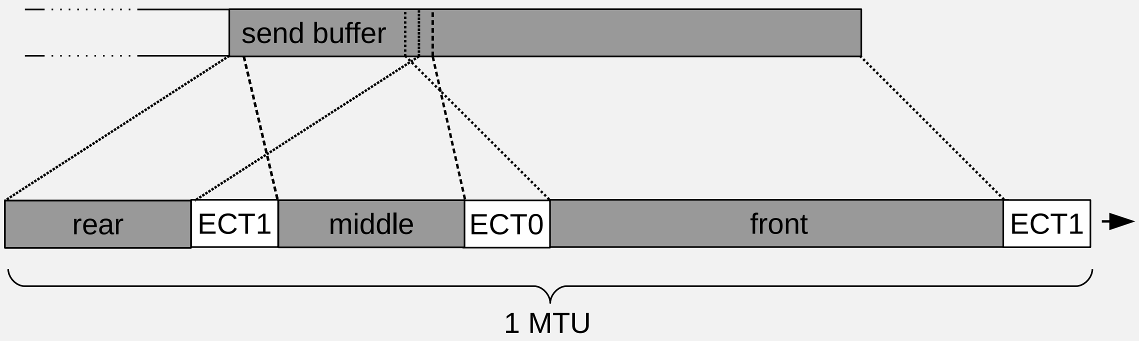


Issue #16: Fall-back to Reno-Friendly on Classic ECN bottleneck

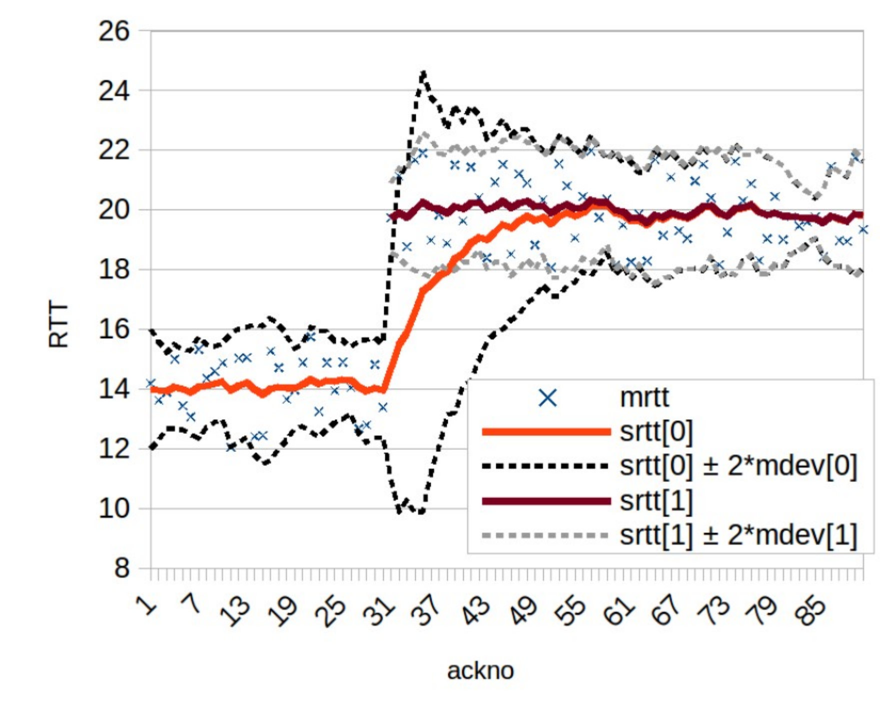
- Passive detection algorithm
  - delayed start following first CE mark
  - 3 weighted elements to detect classic queue
    - mean deviation of the RTT (mdev in TCP)
    - mean Q depth (solely positive factor – min RTT unreliable)
    - degree of self-limiting (app-limited, rwnd-limited) (solely negative 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

Req#1. Scalable Congestion Signalling

> v : number of congestion signals per round trip
> W: congestion window
> p: dropping or marking probability

- congestion signalling is scalable if $v \geq v_0$ (1)
  where v_0 is a reasonable min
- v = (segments per RTT, W) * (probability each will be marked, p)

  $v = Wp$

  substitute in scalability constraint (1)

  $W \geq v_0 / p$ (2)
- can easily derive constraint on steady-state TCP equations from this...

  General congestion control formula: ,
- To satisfy (2), $B \geq 1$

	B
Reno	½
Cubic	¾
DCTCP (prob. AQM)	1
DCTCP (step AQM)	2

Req#2: Limited RTT-dependence

- We have lived with this. Why change?
- Bufferbloat has cushioned us from the impact of RTT-dependent CC
- Low queuing delay leads large RTT flows to starve

Note: this is an anti-starvation requirement not a strong 'fairness' requirement

E.g: base RTT ratio $R_1/R_2 = 200/2 = 100$			
	Qdelay q	Total RTT imbalance $(R_1+q)/(R_2+q)$	
Drop tail	200 ms	$\frac{(200+200)}{(2+200)}$	≈ 2
PIE AQM	15 ms	$\frac{(200+15)}{(2+15)}$	≈ 13
L4S AQM	500 µs	$\frac{(200+0.5)}{(2+0.5)}$	≈ 80

Tension between Reqs 1 & 2

- Scalable congestion signalling $pW \geq v_0$
- Limited RTT-dependence (pW/R const) $pW \propto R$

v : number of congestion signals per round trip
W: congestion window
p: dopping or marking probability
R: Total Round trip time

“Compromise 5” betw Reqs 1 & 2

- signals per RTT

$$pW = \frac{v_0}{lg(R_0/R+1)}$$

scalable signalling

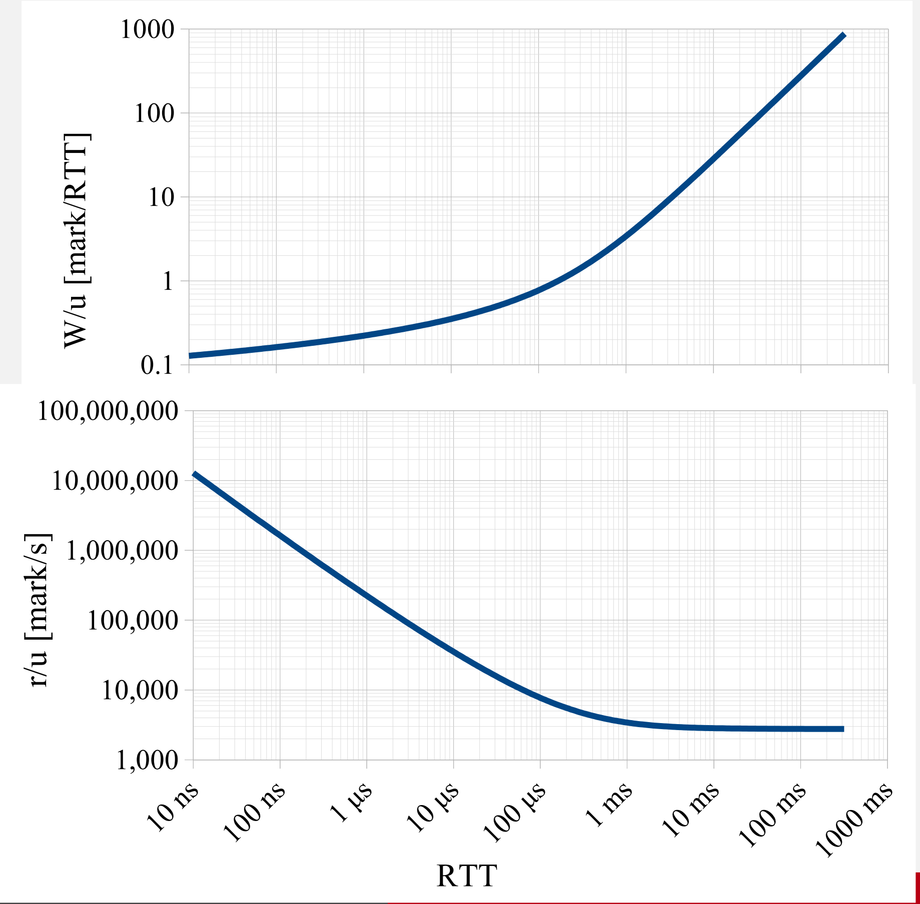
AND

$>>R_0$ RTT-independent
$<<R_0$ not RTT-dependent

- flow rate

$$\frac{pW}{R} = \frac{v_0}{R\,lg(R_0/R+1)}$$

sorry for confusing you all: $p \approx 1/u$



more info

- Resolving Tensions between Congestion Control Scaling Requirements, Bob Briscoe (Simula) and Koen De Schepper (Nokia Bell Labs), Simula Technical Report TR-CS-2016-001; arXiv:1904.07605 [cs.NI] (Jul 2017)

Status against Prague L4S requirements (Jul'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			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	

Scale down to fractional window

- Designed, implemented (Linux base stack) and evaluated (Reno & TCP Prague)
    - works smoothly – complex design process, simple code
    - Research prototype
    - Not yet tested with other TCP Prague components
- Masters thesis of Asad Ahmed and open source code
    - link from L4S landing page
- Booked session to present in iccrg at IETF-107
    - brief preview in TCP Prague side meeting on Thu 08:30 (see next)

Low Latency Low Loss Scalable Throughput (L4S)

Q&A