Low Latency Low Loss Scalable Throughput (L4S)

L4S Drafts Status

draft-ietf-tsvwg-l4s-arch-05
draft-ietf-tsvwg-ecn-l4s-id-09
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-106½ Interim, Feb 2020

tsvwg L4S drafts status

- draft-ietf-tsvwg-...
  - l4s-arch-05 just submitted
  - ecn-l4s-id-09 just submitted
  - coupled-dualq-aqm-10 in good shape, with minor ToDo list. Will update shortly
- Across drafts:
  - Scalable Congestion Control definition expressed in terms of invariant recovery time between congestion signals in steady-state
    - previous definition based on response function was the means not the end
  - “Classic”? see Greg's terminology slides later
  - Turned 'hype' into precise statements – pls review again and suggest text

tsvwg L4S draft revisions

- l4s-arch-05 (intended INF)
  - Explained how L4S works with FQ & DualQ better
  - Fixed cross-refs to 'later'
  - Loads of other minor edits
  - <u>Plans</u>: Fixed abstract hype but still needs work – far too long (next few days). Then review pls

- ecn-l4s-id-09 (intended EXP)
  - Loss recovery in time units: MUST → SHOULD
  - MUST … mark ECT(0) packets under the same conditions as it would drop Not-ECT packets [RFC3168] →
    need not mark ECT(0) packets, but if it does, it will do so under the same conditions as it would drop Not-ECT packets [RFC3168]
  - Added requirement and guidance for L4S experiments to monitor for harm to other traffic
  - Loads of other minor edits
  - <u>Plans</u>: (next few days)
    - Not happy with “Recommended-standard-use” (DS) terminology for complementary identifiers (that might not be DS)
    - MUST [SHOULD?] remain responsive to congestion [with fractional window] → will explain dilemma on list
    - Otherwise done. Review pls

Low Latency Low Loss Scalable Throughput (L4S)

L4S & TCP Prague Status
draft-ietf-tsvwg-ecn-l4s-id

Bob Briscoe, Independent <ietf@bobbriscoe.net>

and many others too numerous to list

TSVWG, IETF-106½ interim, Feb 2020

L4S implementation status

- L4S AQMs
  - DualPI2 & FQ_CoDel_Th Linux code stable
    - continuing to test against TCP Prague updates
  - Product/closed source implementations
    - will gather update reports for IETF-107
- Scalable congestion controls
  - Only discussing our reference implementation (TCP Prague) here
  - Will gather update reports on others for IETF-107

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
- Linux TCP Prague as (a) reference implementation

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

Impl'n status against Prague L4S req's (Nov'19)

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

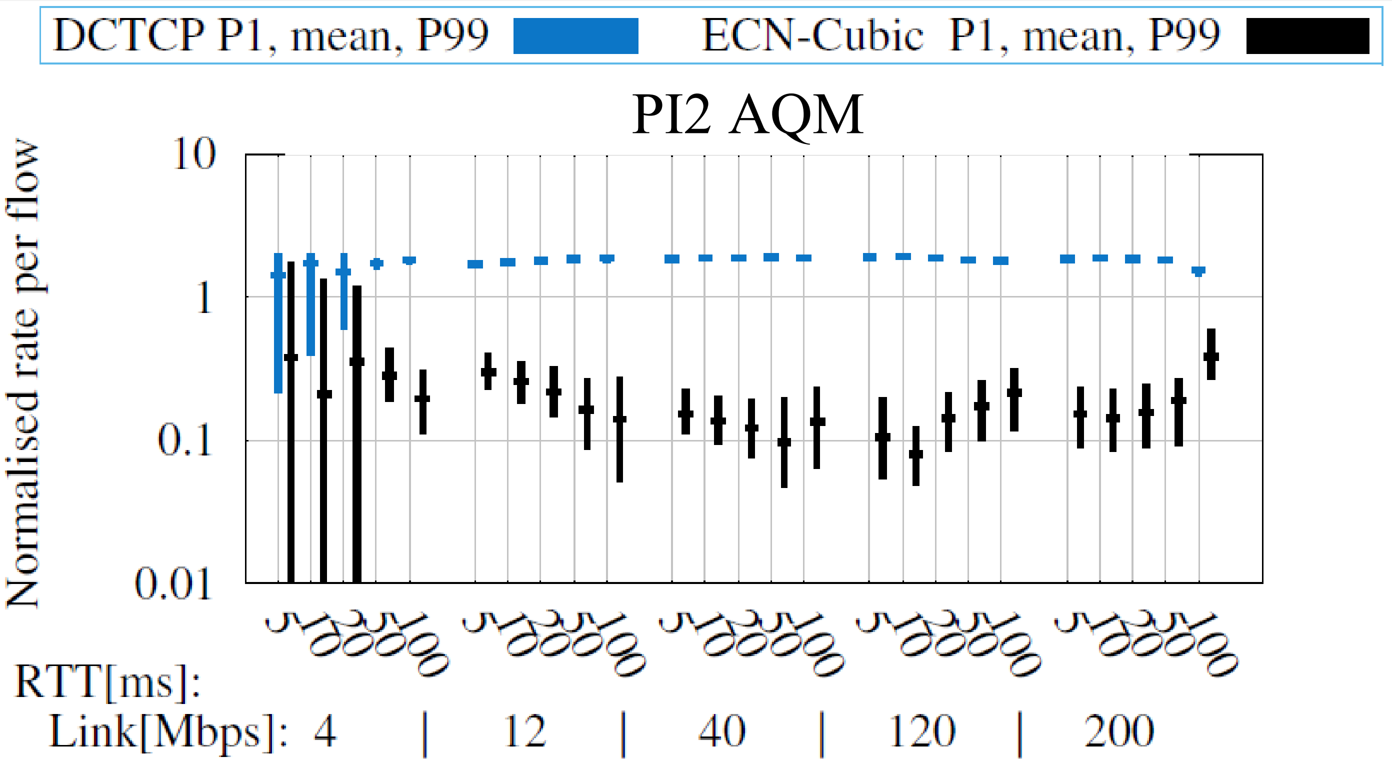
Requirements	base TCP	DCTCP	TCP Prague
L4S-ECN Packet Identification: ECT(1)		**module option**	**mandatory**
Accurate ECN TCP feedback	**sysctl option**	**?**	**mandatory**
Reno-friendly on loss		**inherent**	**inherent**
Reno-friendly if classic ECN bottleneck			evaluat'n in progress
Reduce RTT dependence			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	

Impl'n status against Prague L4S req's (Feb'20)

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

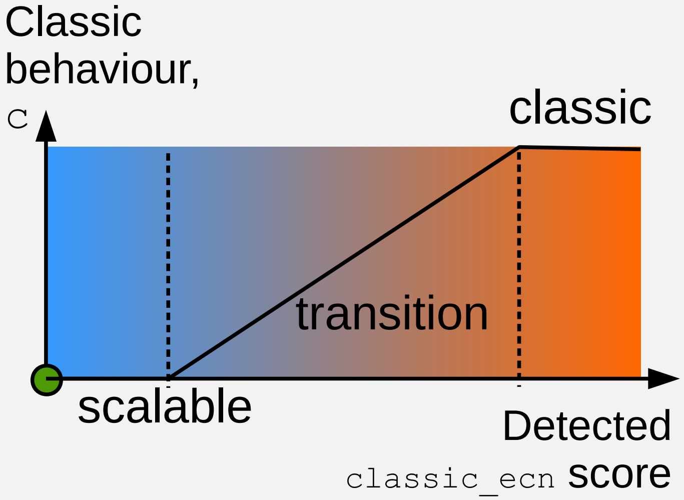
Requirements	base TCP	DCTCP	TCP Prague
L4S-ECN Packet Identification: ECT(1)		**module option**	**mandatory**
Accurate ECN TCP feedback	**sysctl option**	**?**	**mandatory**
Reno-friendly on loss		**inherent**	**inherent**
Reno-friendly if classic ECN bottleneck			evaluat'n in progress
Reduce RTT dependence			evaluat'n in progress
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	

Issue #16: The Problem
RFC3168 ECN AQM in a single Q



Score-Based, not Modal

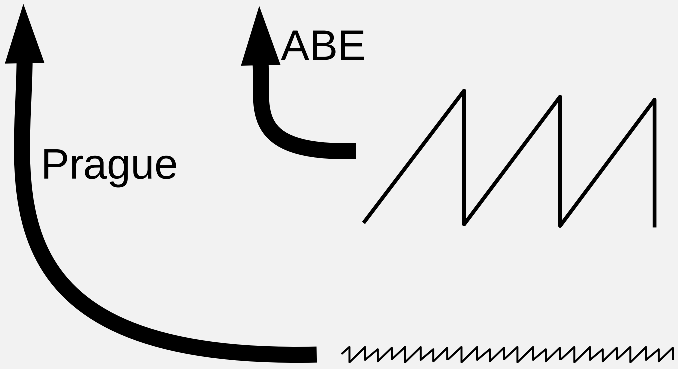
- Detection algorithms – drive a classic ECN AQM score
    1) Passive detection algorithm – primarily based on delay variation
    2) Active detection technique (if passive raises suspicion) – detects different ECT(0/1) treatment
- Detection unlikely to be perfect, so...
    - gradual behaviour change-over from scalable to classic, e.g. TCP Prague becomes Reno
    - hysteresis (sticky) at both ends of spectrum
    - moves faster the more strongly classic is detected
    - but designed to survive transient misleading readings
- Start as scalable (or use per-destination cache)
    - start maintaining score (passively) from first CE mark (but maintain underlying metrics from start of connection)
    - suppresses calculations for short flows (large majority)
    - assumption: classic fall-back only becomes important for longer flows

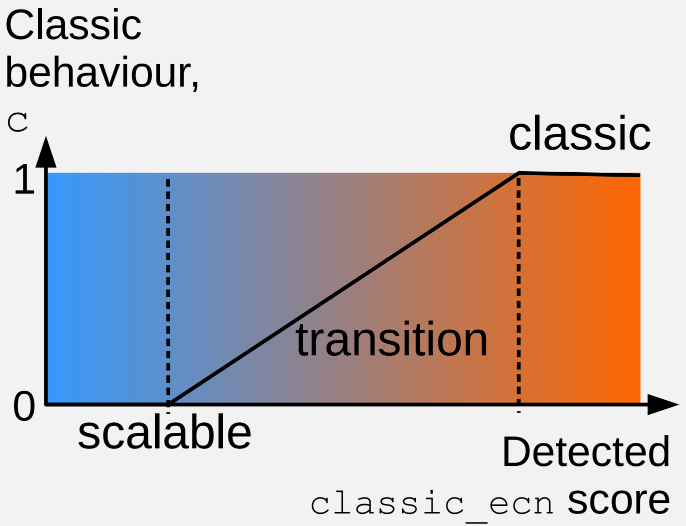


Example: transition from Prague to Reno

```
#define BETA_ABE 0.7                       // ABE: Alternative Backoff with ECN [RFC8511]
#define ALPHA_ABE 2*(1-BETA_ABE)           // 0.6

// On congestion event, reduce ssthresh
reduction = cwnd * max(alpha, c * ALPHA_ABE) / 2;
```

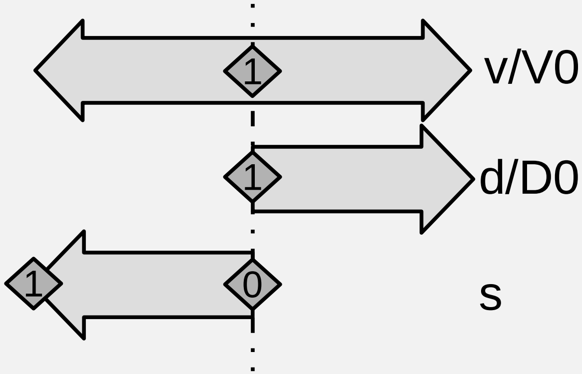




- See discussion paper on design:
  - TCP Prague Fall-back on Detection of a Classic ECN AQM
  - rationale for metrics, pseudocode & analysis

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
    - v, mean deviation of the RTT (mdev in TCP)
    - d, mean Q depth (solely positive factor – min RTT unreliable)
    - s, degree of self-limiting (app-limited, rwnd-limited) (solely negative factor)



- Implemented & Working – see plots/demo
  - in Linux TCP Prague by Asad Ahmed
- Full evaluation in progress – for wide range of conditions

RTT independence in TCP Prague

Olivier Tilmans <olivier.tilmans@nokia-bell-labs.com>

Koen De Schepper <koen.de_schepper@nokia-bell-labs.com>

20-02-2020, TSVWG interim

The throughput of competing AIMD flows depends on their RTT ratio

Queuing delays act as cushion

$$r \sim \frac{1.22}{\sqrt{p} \cdot rtt} \quad \text{or} \quad r \sim \frac{2}{p \cdot rtt}$$

	qdelay	Throughput imbalance
Taildrop	200ms	$\frac{15+200}{.5+200} \sim 1.1$
PIE	15ms	$\frac{15+15}{.5+15} \sim 1.9$
Codel	5ms	$\frac{15+5}{.5+5} \sim 3.6$
L4S AQM	500us	$\frac{15+.5}{.5+.5} \sim 15.5$

Assuming two flows with base RTT of 15ms and 0.5ms, and a constant marking probability

The throughput of competing AIMD flows depends on their RTT ratio

DualQ also gives a different Q per traffic class

$$r \sim \frac{1.22}{\sqrt{p} \cdot rtt} \quad \text{or} \quad r \sim \frac{2}{p \cdot rtt}$$

	Base RTT	Throughput imbalance
DualQ	200ms	$\frac{15 + 200}{.5 + 200} \sim 1.1$
DualQ	15ms	$\frac{15 + 15}{.5 + 15} \sim 1.9$
DualQ	5ms	$\frac{15 + 5}{.5 + 5} \sim 3.6$
DualQ	500us	$\frac{15 + .5}{.5 + .5} \sim 15.5$

Assuming DualQ with targets of 15ms and 0,5ms, equal base RTT and a window-fair coupling (k=2)

New Prague add-on to steer RTT dependence

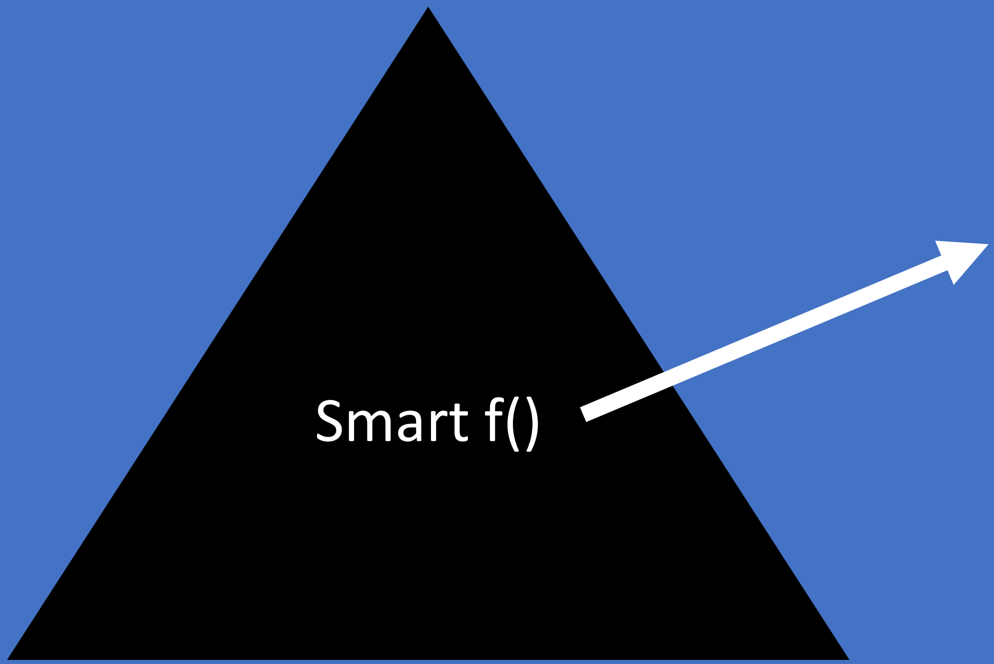
Code to be released soon (demo available)

New Prague CC can have $r \sim \frac{2}{p \cdot f()}$ with a target RTT function $f()$ that can represent any constant or function of flow state

For example $f(rtt) = (rtt + 14.5)$ resulting in:

	Base RTT	Throughput imbalance
DualQ	200ms	$\frac{15 + 200}{.5 + (200 + 14.5)} = 1$
DualQ	15ms	$\frac{15 + 15}{.5 + (15 + 14.5)} = 1$
DualQ	5ms	$\frac{15 + 5}{.5 + (5 + 14.5)} = 1$
DualQ	500us	$\frac{15 + .5}{.5 + (.5 + 14.5)} = 1$

(Long term) Throughput balance
with other RTT flows



IETF or applications
to decide?

Handle
shorter flows
faster

Accelerate faster
at small RTTs

Controlled RTT dependence in TCP Prague

Key changes to TCP Prague

1. We control Additive Increase to behave as a target RTT flow

   Trigger the same amount/frequency of marks as a target RTT flow

2. We leave the Multiplicative Decrease unchanged

   Preserve responsiveness as much as possible to preserve latency

3. Control the EWMA update frequency on the target RTT independently from the e2e RTT

   Ensure that different RTT flows can converge to the same alpha, even on a step

Other changes to TCP Prague

1. Switch to unsaturated marking by default, i.e.,
   cwnd growth is $\sim \frac{1-p}{p}$, regardless of the congestion state (`TCP_CA_CWR`, ...)

   Align to $r \sim \frac{2(1-p)}{p \cdot f()}$ to support unsaturated signal and smoother throughput

2. Generalize fixed-point cwnd manipulation, e.g.,
   carry over remainders from successive cwnd increases and reductions

   The marking probability is usually too low (e.g., 3%) to yield a single packet reduction and the increments can become less than a packet per RTT

Demo/video
f(rtt)= (rtt + 15ms)

Code to be released soon

Accurate ECN feedback in TCP

- Implementation of full tcpm spec
  - by Ilpo Järvinen
    - based on Olivier Tilmans's, based on Mirja Kühlewind's
  - See tcpm list for his design review comments
  - some design tweaks as a result

Immediate Plans: upstreaming to Linux

Faster flow start & faster than additive increase

- Paced Chirping merged into TCP Prague
    - by Joakim Misund
    - over latest Linux kernel
    - default off
    - see previous iccrg talks

<u>Immediate Plans</u>: (re-)engineer research code

Low Latency Low Loss Scalable Throughput (L4S)

Q&A

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)

Passive Classic ECN Bottleneck Detection Algorithm

- 3 weighted elements to detect classic queue
  - v, mean de**v**iation of the RTT (mdev in TCP)
  - d, mean Q **d**epth (srtt – srtt_min)
    (solely positive factor – small srtt_min unreliable)
  - s, degree of **s**elf-limiting (app-limited, rwnd-limited)
    (solely negative factor)
- All metrics already maintained by Linux TCP
  - (?) may need to tune their parameters
- Per-RTT change to classic_ecn score
  - delta = V*lg(v/V0) + D*lg(max(d/D0, 1)) – S*s;
- Constant parameters
  - V, D & S are the weights of each element
  - V0 and D0 are reference values