Low Latency Low Loss Scalable Throughput (L4S)

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

Bob Briscoe, Independent
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

![Graph showing PI2 AQM results for different RTT and Link speeds.](image)

- DCTCP P1, mean, P99
- ECN-Cubic P1, mean, P99

**RTT[ms]:**
- 4
- 12
- 40
- 120
- 200

**Link[Mbps]:**
- 4
- 12
- 40
- 120
- 200

**Normalised rate per flow:**
- 10
- 1
- 0.1
- 0.01
- 0.001
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  - until RFC3168 ECN superseded (or L4S experiment ends)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

- congestion signalling is scalable if $v \geq v_0$ (1) where $v_0$ is a reasonable min
- $v = \text{segments per RTT, } W \times \text{probability each will be marked, } p$
  \[ v = Wp \]
  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), $B \geq 1$

<table>
<thead>
<tr>
<th>$B$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reno</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>Cubic</td>
<td>$\frac{3}{4}$</td>
</tr>
<tr>
<td>DCTCP (prob. AQM)</td>
<td>1</td>
</tr>
<tr>
<td>DCTCP (step AQM)</td>
<td>2</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Qdelay</th>
<th>Total RTT imbalance $(R_1+q)/(R_2+q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drop tail</td>
<td>200 ms</td>
<td>$(\frac{200+200}{2+200}) \approx 2$</td>
</tr>
<tr>
<td>PIE AQM</td>
<td>15 ms</td>
<td>$(\frac{200+15}{2+15}) \approx 13$</td>
</tr>
<tr>
<td>L4S AQM</td>
<td>500 μs</td>
<td>$(\frac{200+0.5}{2+0.5}) \approx 80$</td>
</tr>
</tbody>
</table>
Tension between Reqs 1 & 2

- Scalable congestion signalling \[ pW \geq v_0 \]
- Limited RTT-dependence \((pW/R \text{ const})\) \[ pW \propto R \]

\( v \): 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
  \[ pW = \frac{v_0}{\log(R_0/R+1)} \]

scalable signalling

AND

\[ \gg R_0 \text{ RTT-independent} \]
\[ \ll R_0 \text{ not RTT-dependent} \]

- flow rate
  \[ \frac{pW}{R} = \frac{v_0}{R \log(R_0/R+1)} \]

sorry for confusing you all: \( p \approx \frac{1}{u} \)
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)

<table>
<thead>
<tr>
<th>Requirements</th>
<th>base TCP</th>
<th>DCTCP</th>
<th>TCP Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4S-ECN Packet Identification: ECT(1)</td>
<td></td>
<td>module option</td>
<td>mandatory</td>
</tr>
<tr>
<td>Accurate ECN TCP feedback</td>
<td></td>
<td>sysctl option</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mandatory</td>
<td></td>
</tr>
<tr>
<td>Reno-friendly on loss</td>
<td></td>
<td>inherent</td>
<td>inherent</td>
</tr>
<tr>
<td>Reno-friendly if classic ECN bottleneck</td>
<td></td>
<td></td>
<td>open issue</td>
</tr>
<tr>
<td>Reduce RTT dependence</td>
<td></td>
<td></td>
<td>simulated</td>
</tr>
<tr>
<td>Scale down to fractional window</td>
<td></td>
<td></td>
<td>thesis write-up</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>thesis write-up</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>thesis write-up</td>
</tr>
<tr>
<td>Detecting loss in units of time</td>
<td></td>
<td></td>
<td>mandatory?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimizations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECN-capable TCP control packets</td>
<td></td>
<td>module option off</td>
<td>on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>default off → on later</td>
<td></td>
</tr>
<tr>
<td>Faster flow start</td>
<td></td>
<td></td>
<td>in progress</td>
</tr>
<tr>
<td>Faster than additive increase</td>
<td></td>
<td></td>
<td>in progress</td>
</tr>
</tbody>
</table>

**Linux code:**
- none (simulated)
- research private
- research opened
- RFC
- mainline
### Status against Prague L4S requirements (Nov'19)

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

<table>
<thead>
<tr>
<th>Requirements</th>
<th>base TCP</th>
<th>DCTCP</th>
<th>TCP Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4S-ECN Packet Identification: ECT(1)</td>
<td></td>
<td>module option</td>
<td>mandatory</td>
</tr>
<tr>
<td>Accurate ECN TCP feedback</td>
<td>sysctl option</td>
<td>?</td>
<td>mandatory</td>
</tr>
<tr>
<td>Reno-friendly on loss</td>
<td>inherent</td>
<td>inherent</td>
<td></td>
</tr>
<tr>
<td>Reno-friendly if classic ECN bottleneck</td>
<td></td>
<td></td>
<td>evaluat'n in progress</td>
</tr>
<tr>
<td>Reduce RTT dependence</td>
<td></td>
<td></td>
<td>research code</td>
</tr>
<tr>
<td>Scale down to fractional window</td>
<td>research code</td>
<td>research code</td>
<td>research code</td>
</tr>
<tr>
<td>Detecting loss in units of time</td>
<td>default RACK</td>
<td>default RACK</td>
<td>mandatory?</td>
</tr>
</tbody>
</table>

**Optimizations**

<table>
<thead>
<tr>
<th>ECN-capable TCP control packets</th>
<th>module option off</th>
<th>on</th>
<th>default off → on later</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster flow start</td>
<td>in progress</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faster than additive increase</td>
<td>in progress</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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