Some Congestion Experienced

draft-morton-tsvwg-sce-01

draft-morton-tsvwg-cheap-nasty-queuing-01

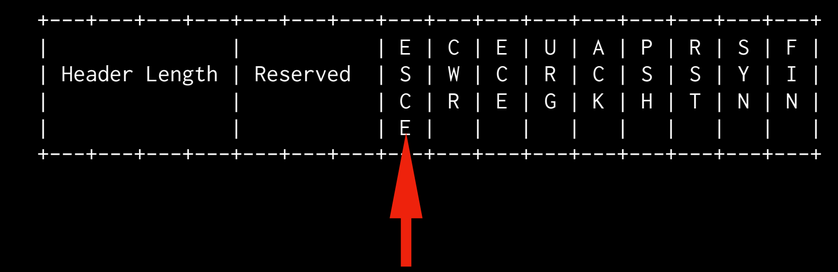
Jonathan Morton

Pete Heist

Rodney W Grimes

Some Congestion Experienced

Binary	Keyword	References
00	Not-ECT (Not ECN-Capable Transport)	[RFC3168]
01	SCE (Some Congestion Experienced)	[This Internet-draft]
10	ECT (ECN-Capable Transport)	[RFC3168]
11	CE (Congestion Experienced)	[RFC3168]



- Redefines ECT(1) as SCE, a high-fidelity congestion signal.
- Retains all other RFC-3168 details, for full backwards compatibility.
- Uses the former NS bit as ESCE, for TCP feedback.
- Multiple instances of running code available!

SCE Design Philosophy

“First, do no harm.”

Hippocrates

“Heterogeneity is inevitable and must be supported by design.”

RFC-1958 § 3.1

“Effective congestion control is REQUIRED.”

RFC-8311 § 5.2.1

“Simplify, then add lightness.”

Colin Chapman, Lotus Cars

SCE is a Signalling Mechanism

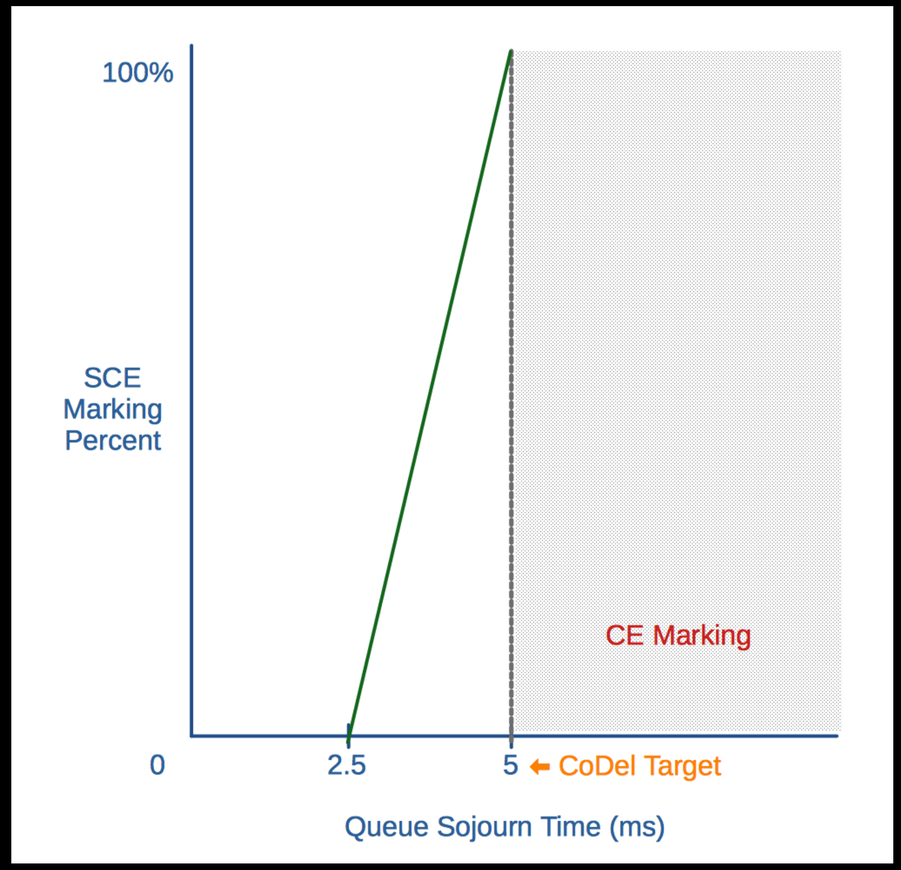
- RFC-3168 ECN:
  - Information flows primarily from network to endpoints.
  - Packets are marked ECT by senders, to inform network nodes that they may mark packets (with CE) instead of dropping them, as an explicit congestion signal.
  - Receiver converts CE signal into reliable, imprecise ECE signal.
- Some Congestion Experienced:
  - Additional information channel from network to endpoints.
  - Receiver converts SCE signal into unreliable, high-fidelity ESCE.

SCE Backwards Compatibility

- Normal AIMD principles apply
  - Normal cwnd growth rate (à la Reno, CUBIC).
  - Response to loss is TCP friendly.
  - Response to CE marks is RFC-3168 compliant (as amended).
- Existing RFC-3168 middlebox AQMs treat SCE marks as ECT
  - Can still mark them with CE; compatibility is good.
- Existing endpoints ignore SCE marks and ESCE bit
  - Transparent fallback to RFC-3168.
- Meaning of CE preserved
  - SCE can be a soft "cruise control" signal.
  - Key advantage over DCTCP.

SCE: Signal Network ⇛ Endpoints

- High-fidelity congestion control signal
  - Many marks per RTT, versus many RTTs per mark, in steady state.
- Easily added to existing AQM algorithms
  - Easiest if FQ is also implemented.
  - Experimentation with marking strategies possible.
  - Ideally, use a burst-tolerant strategy (eg. Codel).
- RFC-3168 CE marks still relevant for:
  - Abrupt reductions in path capacity.
  - Backwards compatibility.



SCE Endpoint Responses

- Receiver: echo SCE marks using ESCE bit in acks.
  - This is an accurate, non-reliable mechanism.
  - As much as 100% relative error in feedback is tolerated without fundamentally damaging control loop stability.
  - Non-SCE receivers simply leave ESCE bit cleared.
- Sender: respond to ESCE feedback by reducing cwnd slightly.
  - DCTCP: reduce cwnd by half ESCE-flagged data.
  - ELR: reduce cwnd by sqrt(cwnd segs) * ESCE-flagged data.
  - Non-SCE senders simply ignore ESCE flag.

SCE & the Prague Requirements

1: Packet Identifier

- SCE identifies itself as RFC-3168 compatible, with ECT(0).

2: Accurate ECN Feedback

- CE feedback is reliable, but not accurate, as per RFC-3168.
- SCE feedback is accurate, but not reliable, using ESCE flag.
  - See draft-grimes-tcpm-tcpsce-01 and tomorrow's TCPM talk.

SCE & the Prague Requirements

3: Fallback to MD on Loss

- Yes.

4: Fallback to MD on RFC-3168 mark

- Yes: CE is treated unambiguously as per RFC-3168.
- High-fidelity ECN information is carried only by new SCE mark.

SCE & the Prague Requirements

5: Reduce RTT Dependence

- Whenever steady-state cwnd is determined solely by p(mark):
  - Single-queue self-competition with diverse RTTs will converge to same cwnd in each flow.
  - Therefore, throughput inversely proportional to RTT.
  - This covers Reno, DCTCP, & current SCE implementations.

- CUBIC under RFC-3168 does better than this! (Oops.)
  - CUBIC steady-state cwnd function includes $RTT^{3/4}$ term.

SCE & the Prague Requirements

6: Scale down to fractional cwnd

- We take this to mean being capable of reducing send rate below 1*MSS/RTT, by spacing segment transmissions…
  - Yes, using sub-unity TCP pacing scale factors.
- …and/or reducing segment size.
  - We don't do this. If a use case shows up, we might try it.

7: Reordering tolerance on time basis

- Yes, inherited from RACK-capable Linux TCP stack.

SCE Progress Since Montreal

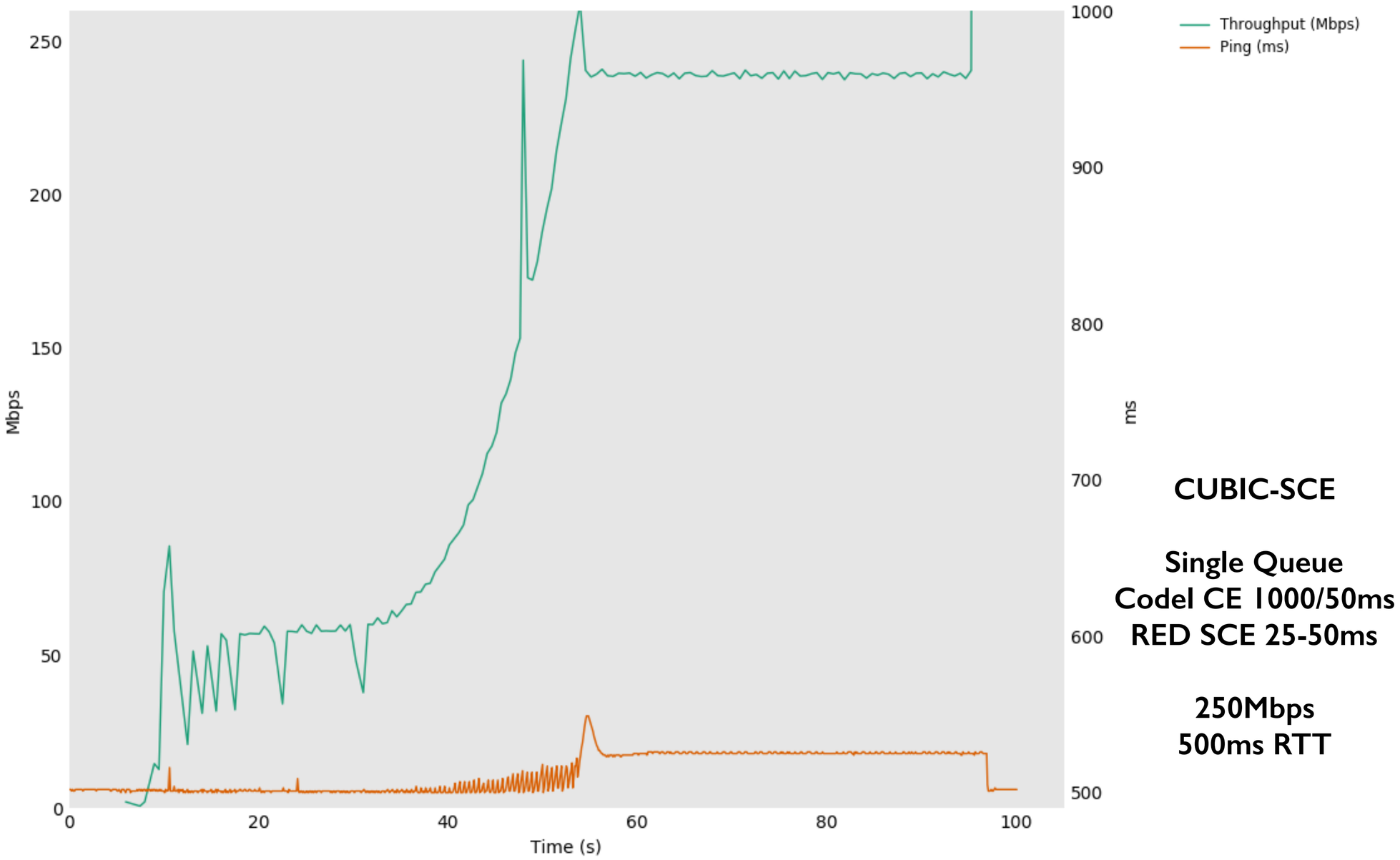
- CNQ (Cheap Nasty Queuing) implementation addresses SCE-AQM deployment where full FQ is infeasible.
  - NB: plain old RFC-3168 single-queue AQMs also behave well on SCE flows.
- COBALT 2 addresses bursty traffic, using Codel marking strategy.
  - Tested this at the Hackathon this weekend.
- CUBIC-SCE scales up to LFNs much better than Reno-SCE.
  - Still an area of research.
- Evaluated SCE according to Prague Requirements.
  - We seem to meet all but #5, in which we're no worse than Reno.
- Began evaluation of SCE according to RFC-5033.
  - This is likely to take a while.

CUBIC-SCE

- Well-known CUBIC algorithm scales well to high BDPs.
- Start with standard codebase, add SCE response:
  - Shift cubic curve towards level inflection point…
    - Reduces growth rate if in super-linear growth regime.
  - …and apply SCE's usual ELR reduction to cwnd & cubic curve.

```
reno_accum -= acked_bytes * sqrt(cwnd);
if(reno_accum <= -(cwnd * mss)) {
  reno_accum += cwnd * mss;
  cwnd--;
}
```

- The code is in the usual repo.



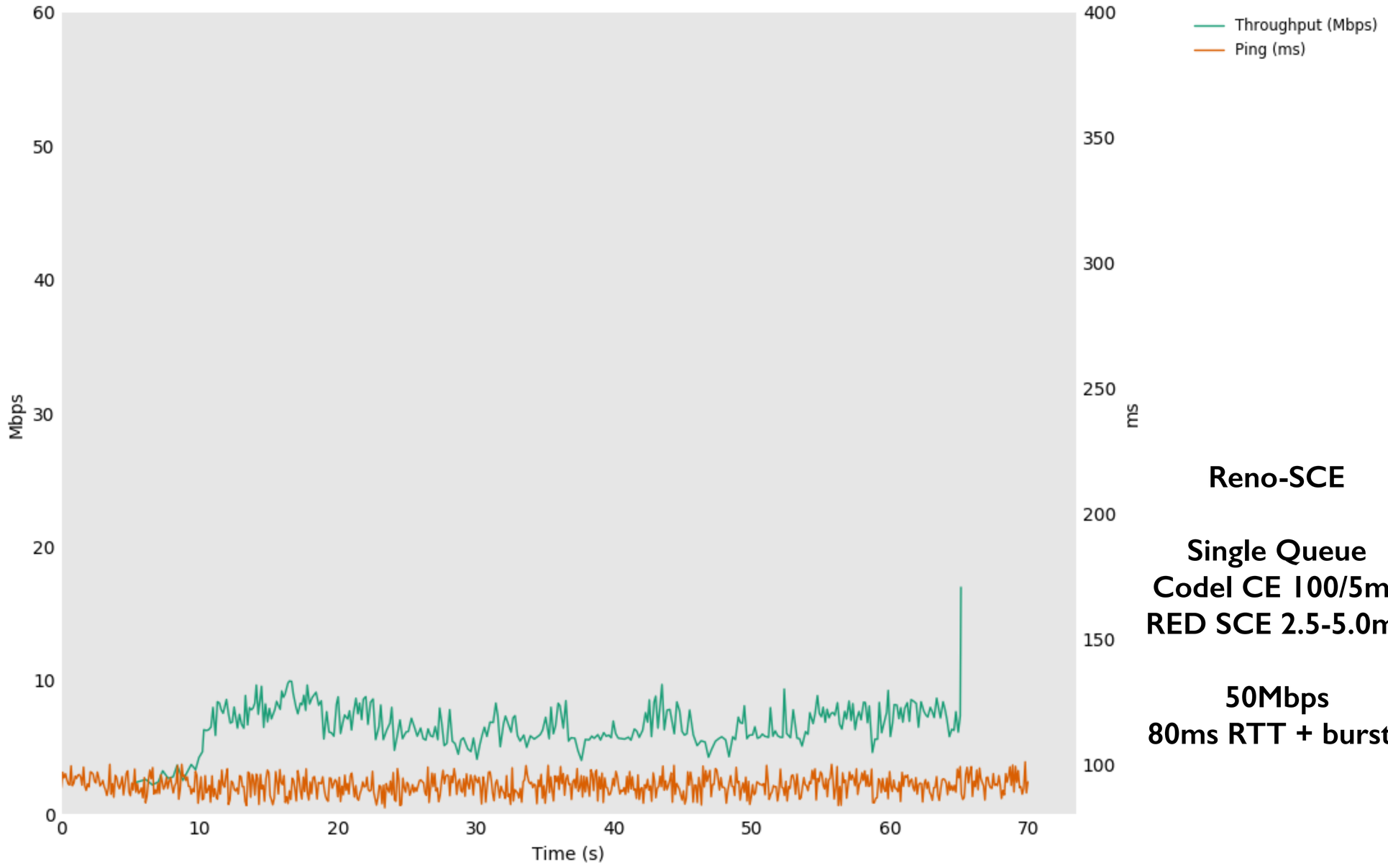
CUBIC-SCE

Single Queue
Codel CE 1000/50ms
RED SCE 25-50ms

250Mbps
500ms RTT

COBALT 2

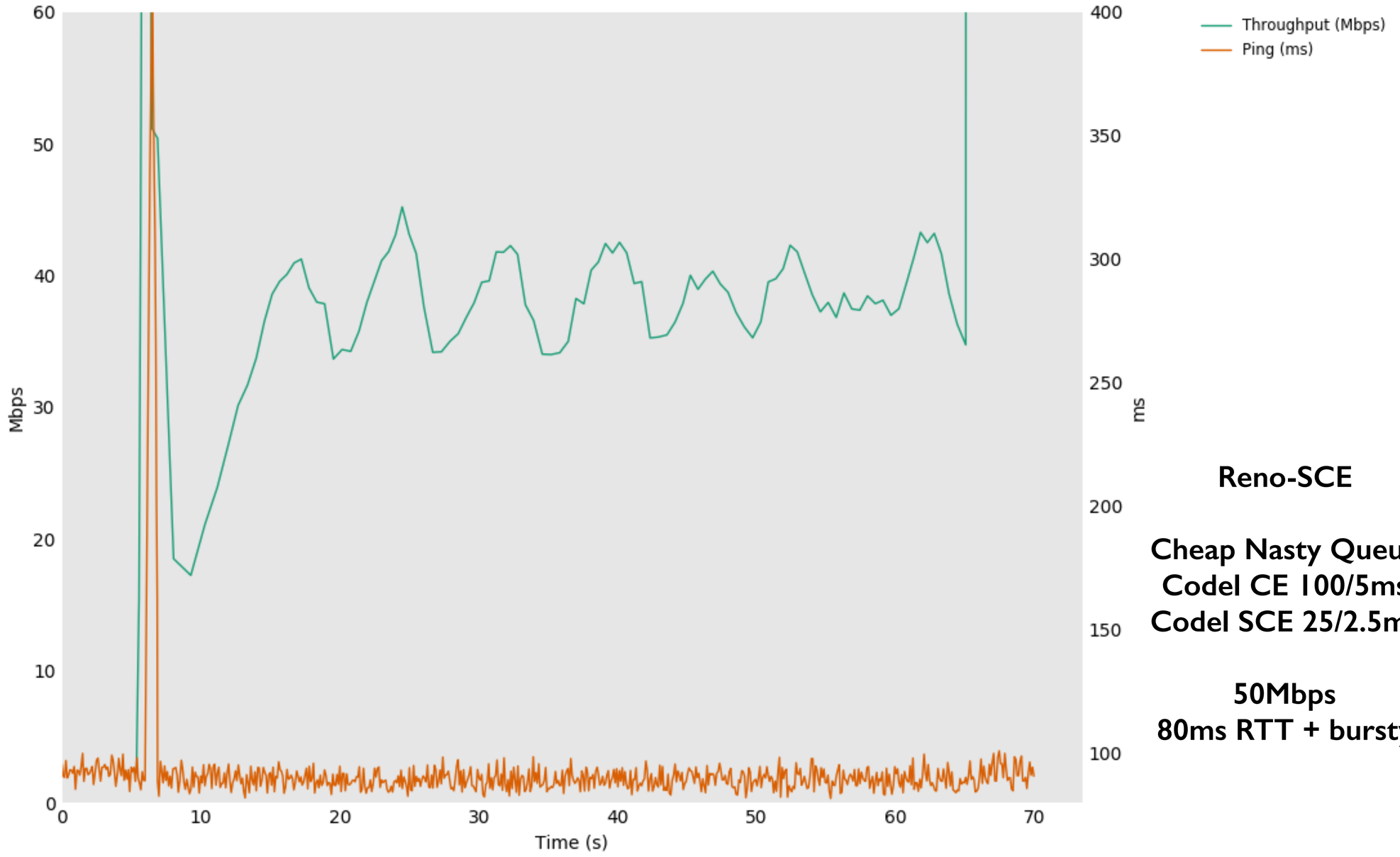
- Initial SCE marking strategies were threshold or RED ramp.
  - Good for initial work, but suffers on bursty links.
  - Unnecessary congestion signals produced when burstiness exceeds queue depth at which marking begins.
- We are now trying a second Codel instance for SCE marking.
  - Codel has a built-in burst tolerance.
  - Hackathon: try 10ms delay variation & 32-packet bursts (like wifi)
    - SCE-marking Codel with 25ms interval, 2.5ms target.
    - CE-marking Codel instance at default 100ms/5ms.
    - Dropping BLUE instance triggers at 400ms overload.



Reno-SCE

Single Queue
Codel CE 100/5ms
RED SCE 2.5-5.0ms

50Mbps
80ms RTT + bursty



Reno-SCE

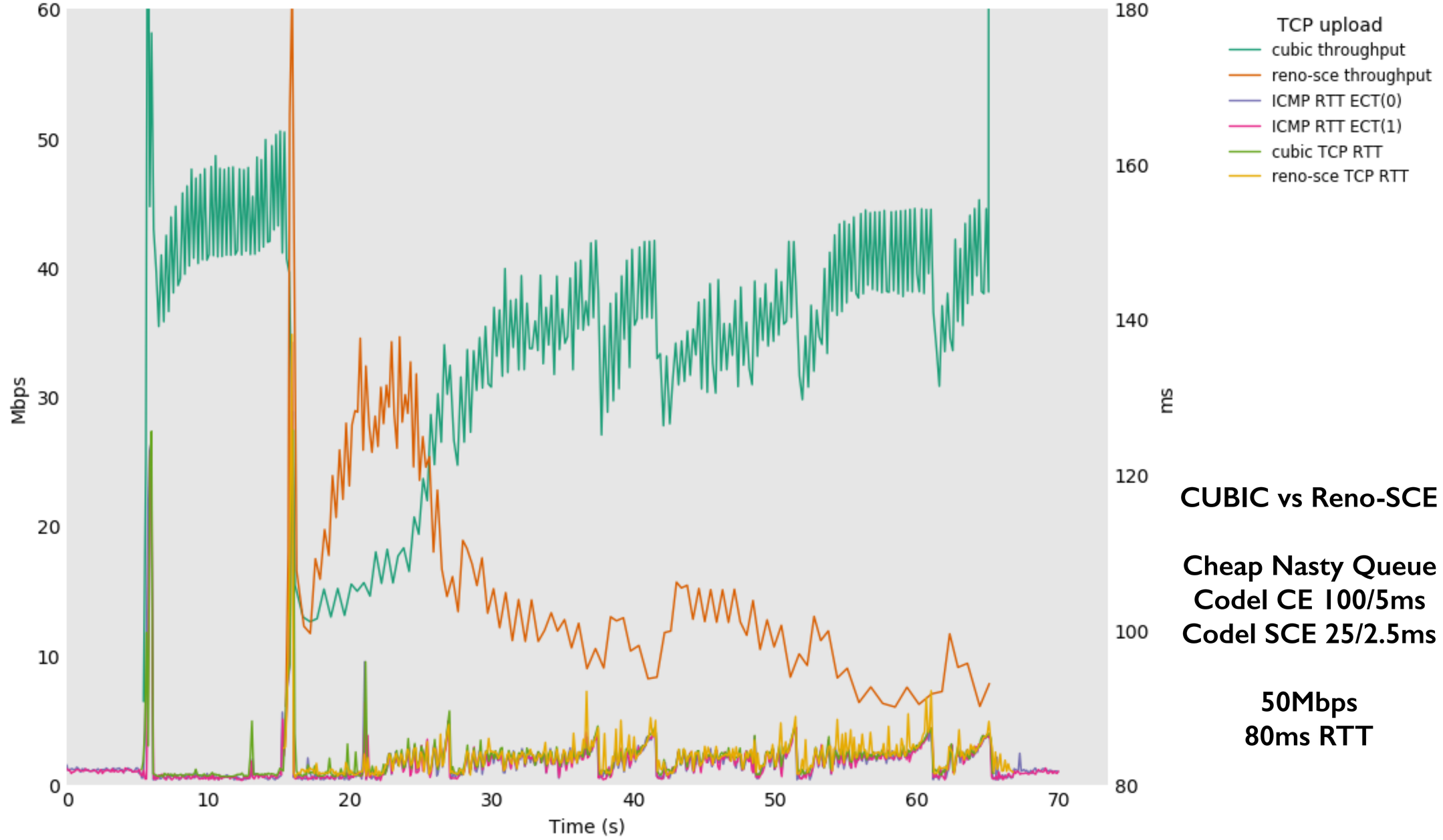
Cheap Nasty Queue
Codel CE 100/5ms
Codel SCE 25/2.5ms

50Mbps
80ms RTT + bursty

Cheap Nasty Queuing

We prefer FQ. However…

- There are places where deploying FQ might be difficult, but deploying a good AQM with SCE support is desirable.
- In a single queue SCE-AQM, SCE flows yield to RFC-3168 flows.
- CNQ implements a limited form of FQ with minimal complexity.
  - Two FIFOs in total, one counter per flow bucket, one AQM.
  - In many situations, CNQ behaves like a FIFO with AQM.
- If the AQM maintains queue depth at 5ms:
  - Smooth up-to 2.5Mbps flows are considered sparse, prioritised.
  - This puts a useful floor on any potential starvation.
- All flows kept in FIFO order.



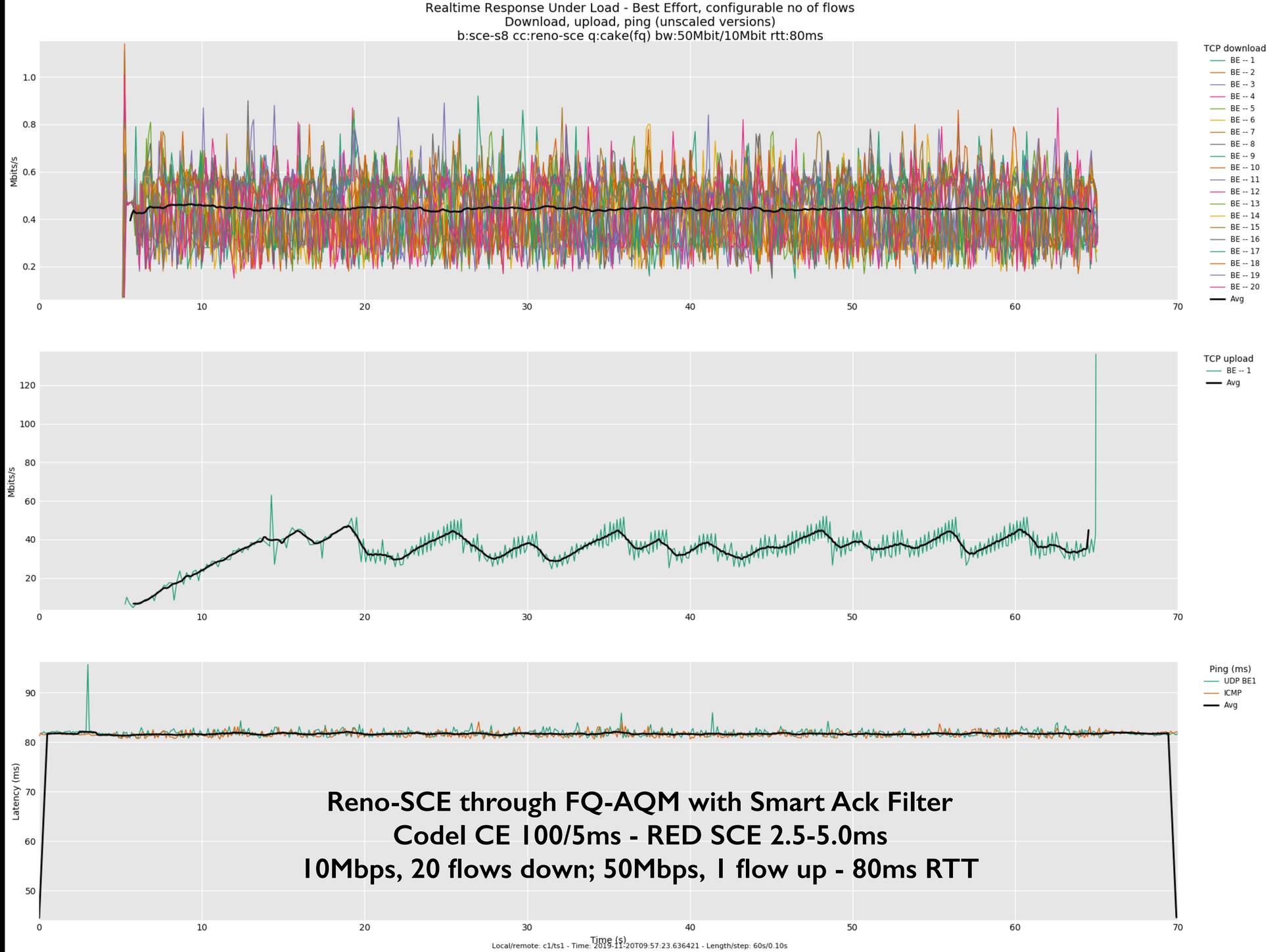
CUBIC vs Reno-SCE

Cheap Nasty Queue
Codel CE 100/5ms
Codel SCE 25/2.5ms

50Mbps
80ms RTT

SCE: Next Steps

- Requesting WG adoption of draft-morton-tsvwg-sce.
  - We believe SCE is a useful experiment that is already safe to deploy in the general Internet, and we're happy to back up that assertion with data.
- Some comments on the -01 drafts have been received & noted.
- Continue RFC-5033 evaluation, will update draft to suit results.
- CUBIC-SCE implementation has some bugs to track down.
- Research methods of meeting Prague Requirement #5.
  - SCE framework permits flexibility in both marking strategies and sender responses, thus facilitating such research without compromising safety.


Realtime Response Under Load - Best Effort, configurable no of flows
Download, upload, ping (unscaled versions)
b:sce-s8 cc:reno-sce q:cake(fq) bw:50Mbit/10Mbit rtt:80ms
TCP download
BE -- 1
BE -- 2
BE -- 3
BE -- 4
BE -- 5
BE -- 6
BE -- 7
BE -- 8
BE -- 9
BE -- 10
BE -- 11
BE -- 12
BE -- 13
BE -- 14
BE -- 15
BE -- 16
BE -- 17
BE -- 18
BE -- 19
BE -- 20
Avg
Mbits/s
TCP upload
BE -- 1
Avg
Mbits/s
Ping (ms)
UDP BE1
ICMP
Avg
Latency (ms)
Reno-SCE through FQ-AQM with Smart Ack Filter
Codel CE 100/5ms - RED SCE 2.5-5.0ms
10Mbps, 20 flows down; 50Mbps, 1 flow up - 80ms RTT
Time (s)
Local/remote: c1/ts1 - Time: 2019-11-20T09:57:23.636421 - Length/step: 60s/0.10s

SCE + L4S: Coexistence

- If SCE is adopted by the WG, there may be two experiments operating on the same set of ECN-field codepoints and TCP header flags simultaneously, with diverging uses.
- We believe the consequences of mixing the two experiments are reasonably benign:
  - L4S depends on AccECN, which SCE does not negotiate, so endpoints will not directly conflict in interpreting codepoints on the same flow.
  - L4S traffic running through an SCE AQM will behave like RFC-3168.
  - SCE traffic running through an L4S AQM will look mostly like "Classic ECN" traffic. The exceptions should be tolerated acceptably well by SCE endpoints, which will also back off more conservatively than L4S expects from CE marking.
- SCE functionality in endpoints and AQMs is off-by-default and is easy to turn off if required. This provides for experiment termination.

Some Congestion Experienced

Any questions?