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)

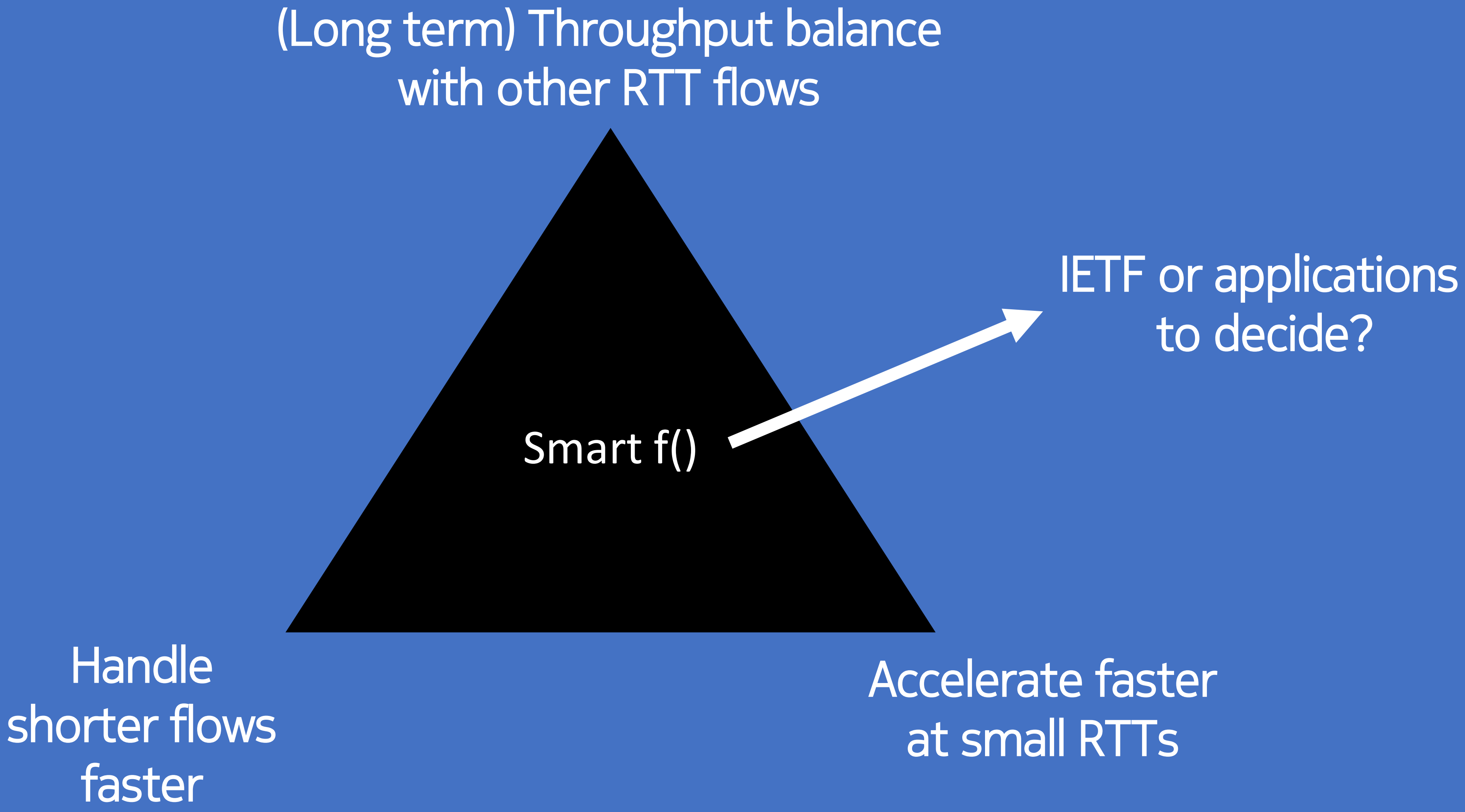
New Prague add-on to steer RTT dependence

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?
Smart f()
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

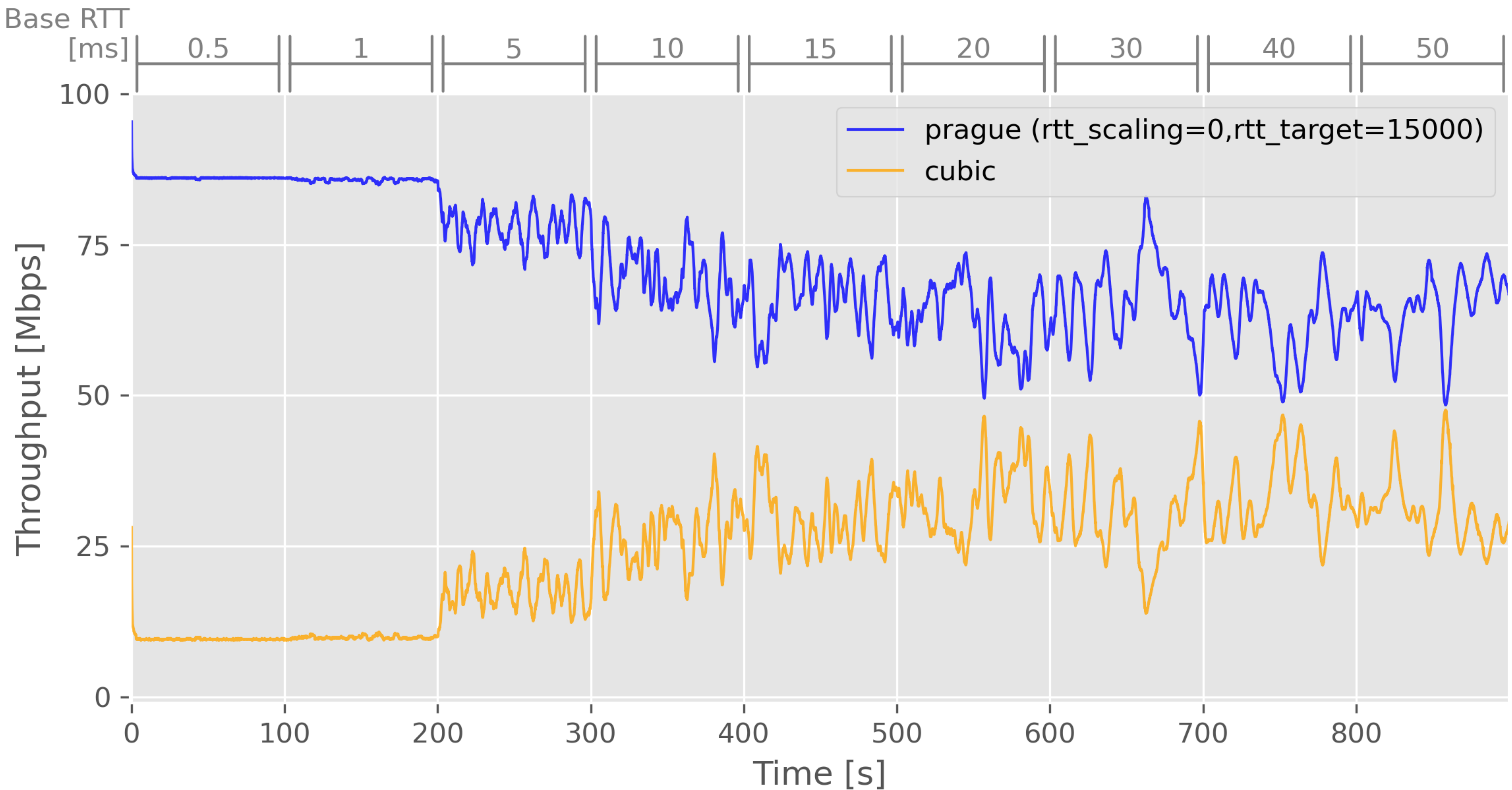
   The marking probability is usually too low (e.g., 3%) to yield a single packet reduction
   and the increments can become less than a packet per RTT

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

Code available in https://github.com/L4STeam/linux
RTT dependence can be controlled with the prague_rtt_* module parameters, e.g.,
`echo 3 | sudo tee /sys/module/tcp_prague/parameters/prague_rtt_scaling`

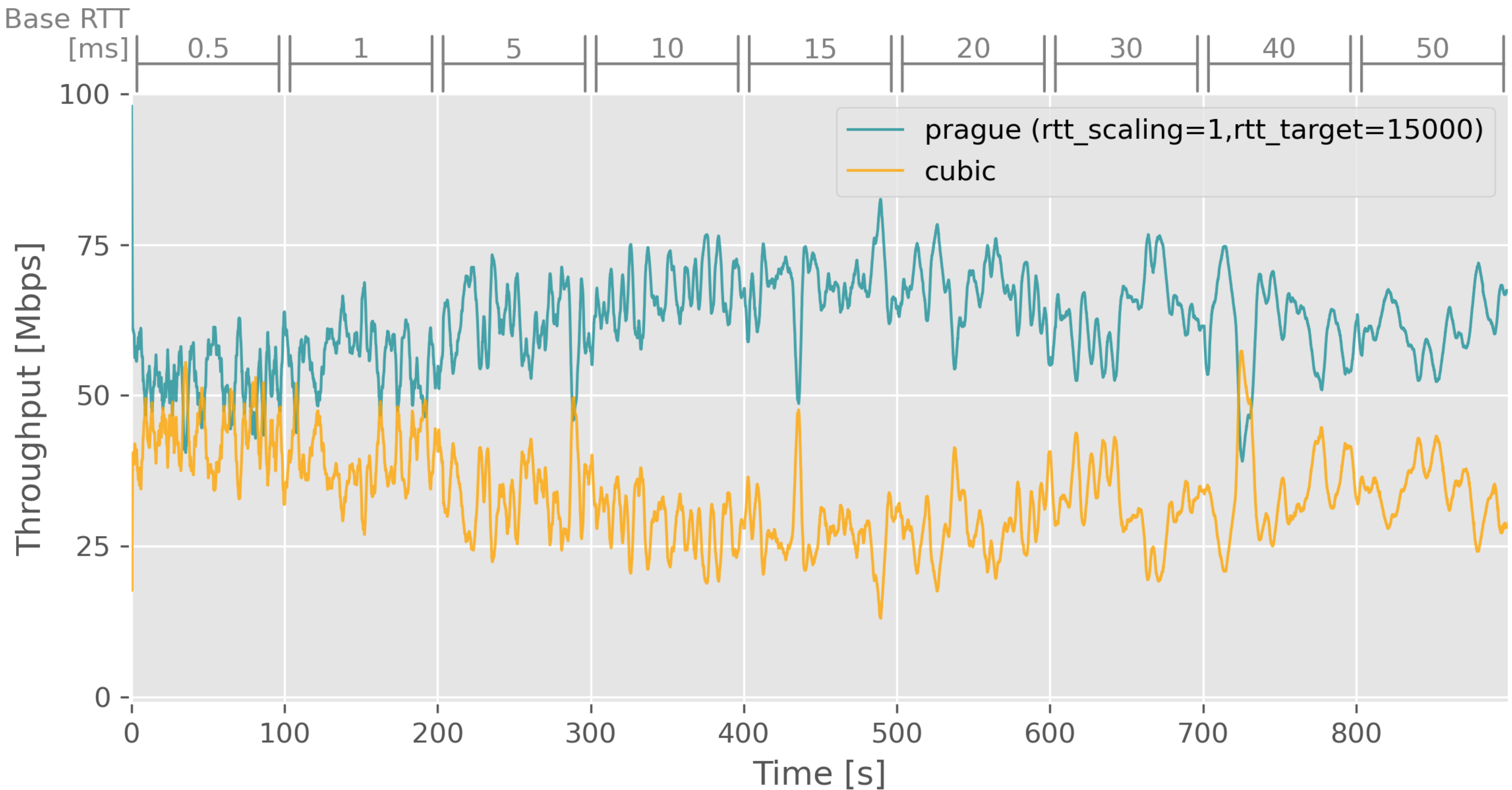
Test results—100Mbit/s bottleneck, increasing base RTTs

Baseline—Prague vs Cubic



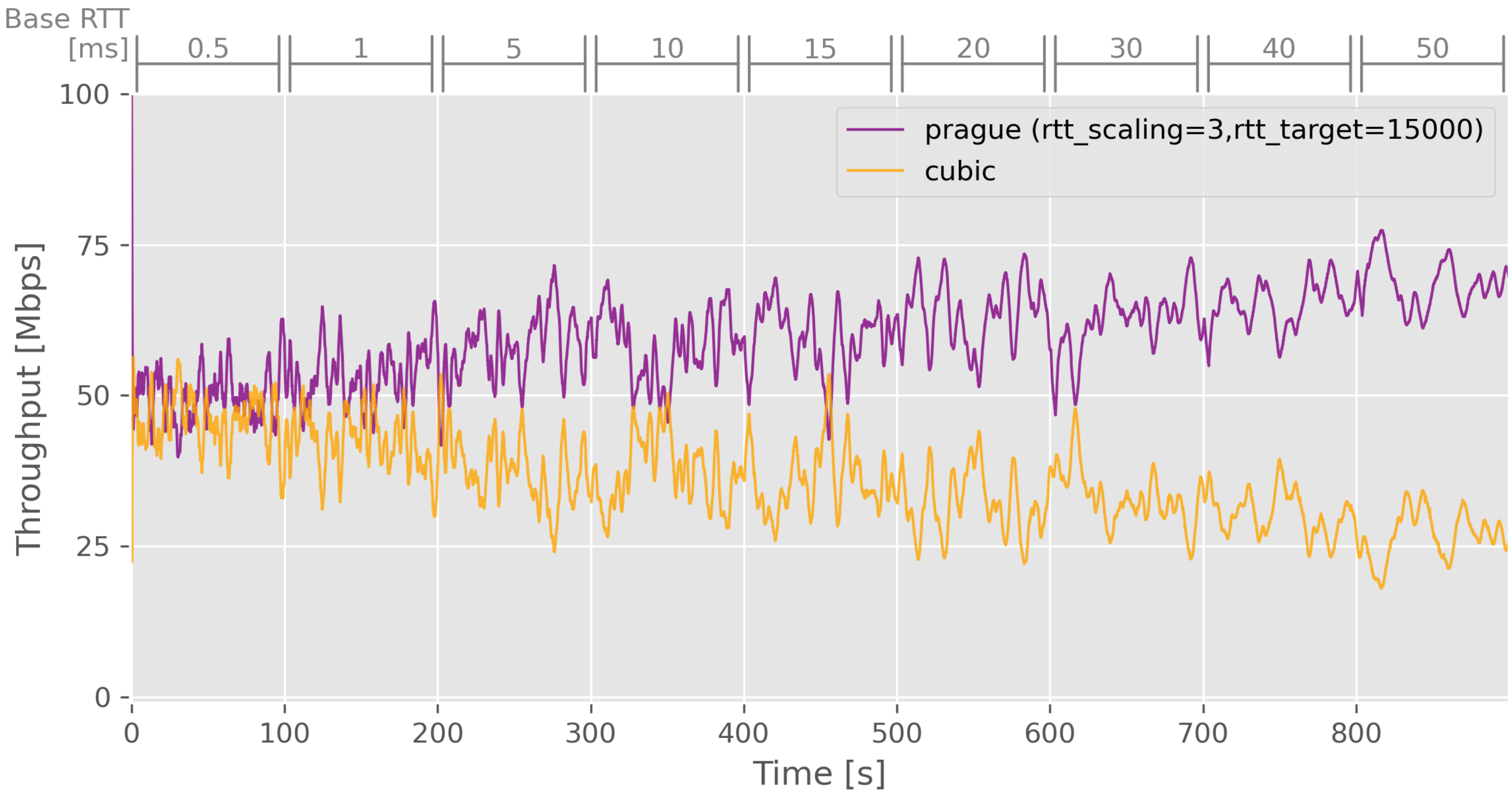
Test results—100Mbit/s bottleneck, increasing base RTTs

Prague [f(rtt) = max(15ms, rtt)] vs Cubic



Test results—100Mbit/s bottleneck, increasing base RTTs

Prague [f(rtt) = rtt + 15ms] vs Cubic



Test results—100Mbit/s bottleneck, mixed base RTTs

Prague [f(rtt) = max(15ms, rtt)] vs Prague