University of Glasgow

Measurement-Based TCP Parameter Tuning in Cloud Data Centers

Simon Jouet

University of Glasgow

[s.jouet.1@research.gla.ac.uk]

Background TCP Congestion Control

"For a transport endpoint embedded in a network of **unknown topology** and with an **unknown**, **unknowable** and constantly changing population of **competing conversations**, only one scheme has any hope of working –exponential backoff-"

Congestion Avoidance and Control, Van Jacobson, 1988

[…] a WSC server is deployed in a relatively **well-known environment**, l**eading to possible optimizations for increased performance**. […] lower packet losses than in long-distance Internet connections. Thus we **can tune transport** or messaging parameters (timeouts, window sizes, etc.) for **higher communication efficiency**.

The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Luiz André Barroso, Urs Hölzle, 2009

TCP Connection Parameters are static

TCP Congestion control parameters are based around default values optimized for Long Fat Pipes (LFP) / Wide Area Network (WAN)

- *Minimum and Initial Retransmission Timeout, 200ms and 3s*
- *Initial congestion window, 10 segments (multiple of MSS)*

However since 80% of DC traffic stay inside cloud DC, shouldn’t the traffic be optimized for internal communication ?

Optimize for Low Latency, High throughput environment

- *Lot of flows, small in size*
- *Less than 1 ms RTT for same rack traffic*
- *As much as 10ms RTT for east-west traffic*
- *Gigabit Ethernet*

Throughput Incast Collapse

In many-to-one traffic pattern (MapReduce) many flows share the same egress queue

Packets are dropped when the buffer reach maximum occupancy (tail-drop)

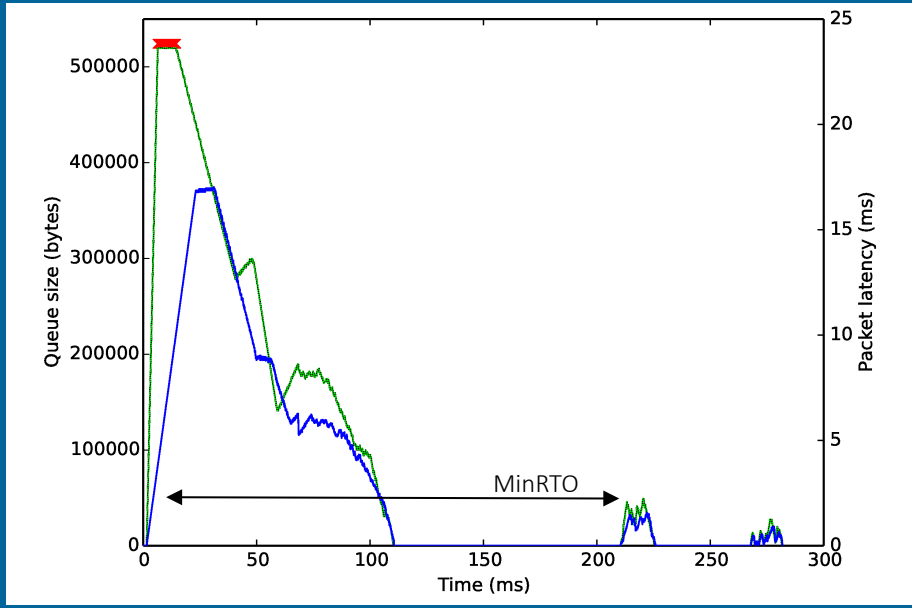
If not enough ACK to trigger F-RTO, wait for retransmit timer timeout

Create burst of traffic separated by long idle period, low overall throughput

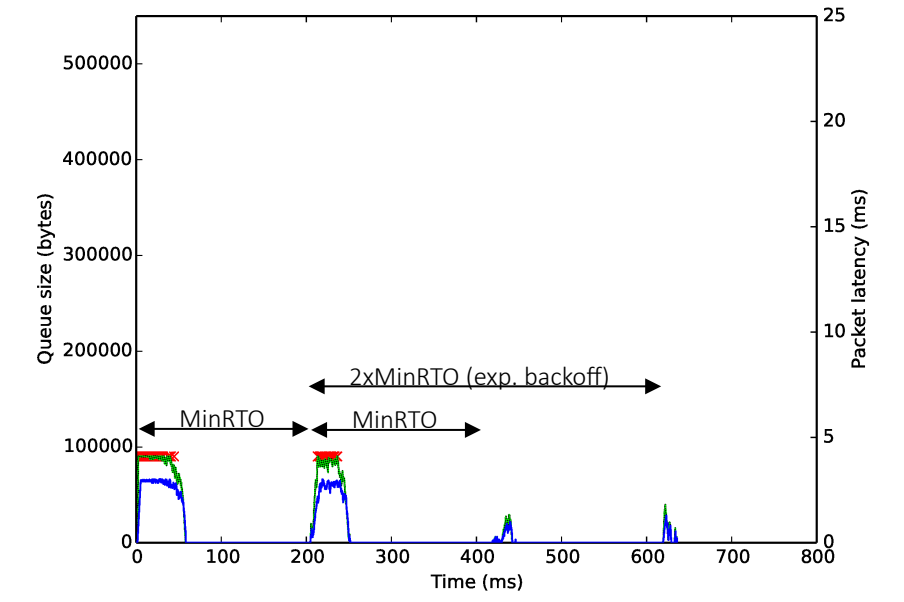
Deep buffers have lower drop rate, high(er) throughput BUT long traversal time

Shallow buffers have a high drop rate, low throughput, short traversal time

Buffering and timeouts



Incast collapse in deep buffered switch



Incast collapse in shallow buffered switch

Buffer Size	Goodput (Mb/s)	Completion time (ms)	Packet drop	Delay avg/max (stddev)	On-Off ratio
512kB	56.94 (45%)	274.42	585	6.0/16.9 (5.1)	1.003
85kB	24.58 (20%)	635.65	1058	1.7/3.0 (1.2)	0.277

Parameter tuning

Configure the TCP parameters based on available network information

Set minRTO to the maximum possible fabric delay

Set congestion window to match the network BDP

Link delay

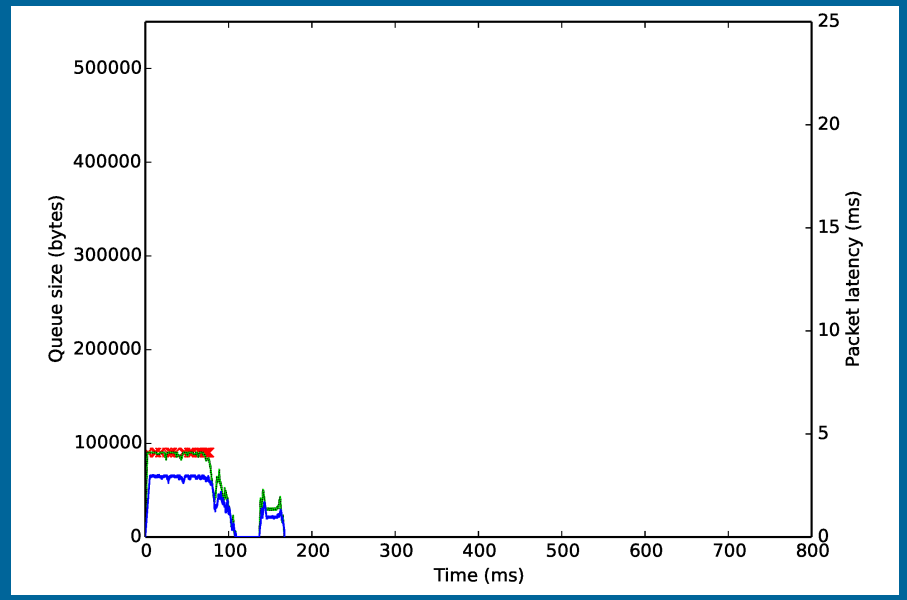
Maximum buffer delay

$$minRTO = \sum_{i=1}^{n} Li + \sum_{i=1}^{n} Bi/Ti$$

Link throughput

$$IW = \min_{i \in R} Ti/Ni \times \sum_{i=1}^{n} Li$$

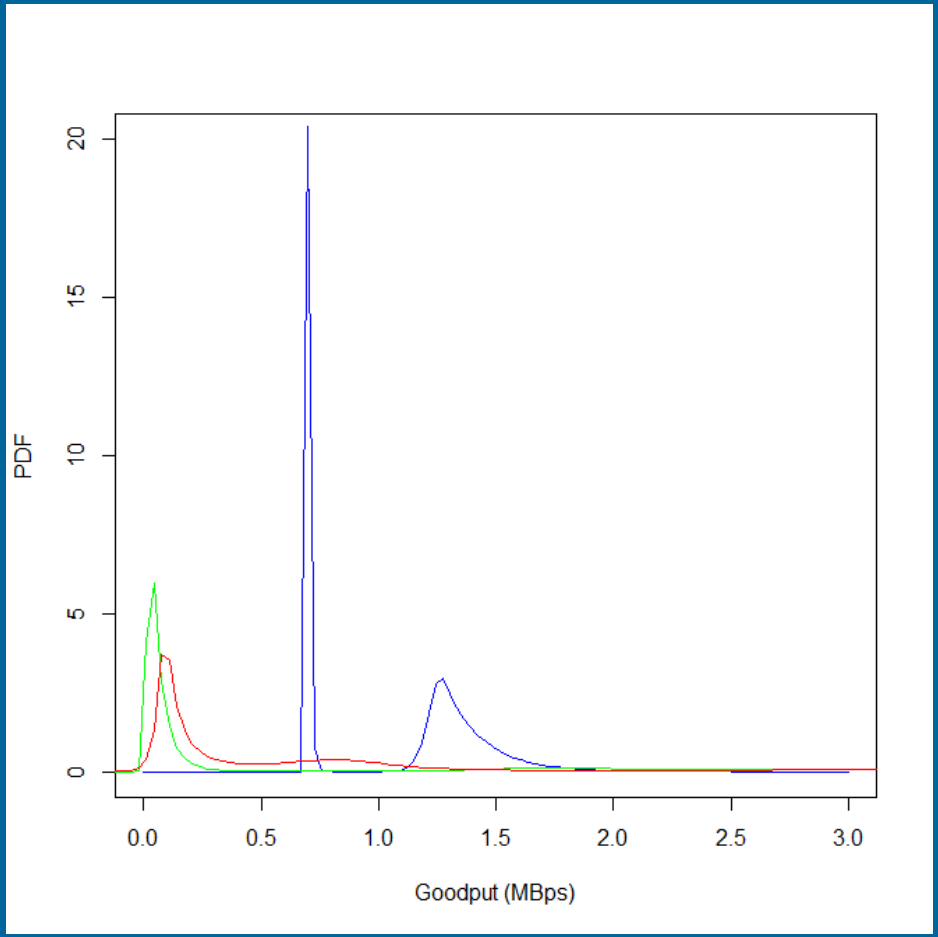
Number of flows



Shallow buffer, minRTO 1ms, Cwnd 1

- 166ms completion time
  - 1.5x faster than DBS and 3.5x SBS
- Packet drop 213
  - 2.7x less than DBS, 5x less than SBS
- Goodput
  - 94.11 Mbps, 1.7x DBS
- Latency
  - average 1.83ms, max 3ms, 1.1 stddev

What about AQM and ECN ?



Explicit Congestion Notification (in green) achieve extremely low per flow goodput as it notifies end-host of congestion after it passed

RED (in red) triggers too many drops creating unfair bandwidth allocation for some flows

Without RED or ECN but tuned TCP stack, higher and more stable goodput.

Conclusion

A lot of information is available in a DC

- topology, latency, throughput
- SDN can provide flow count and flow route

TCP “conservative” parameters are not really conservative for a DC environment

- retransmission timeout is 2—3 order of magnitude too large
- Initial congestion window 1 order of magnitude too large

Applying network information

- Doesn’t need kernel modification
- High throughput, low and stable latency, shorter completion time

Questions ?