A large-scale deployment of DCTCP

NSDI 2024

Prashanth Kannan

Abhishek Dhamija, Balasubramanian Madhavan, Hechao Li, Jie Meng, Lawrence Brakmo, Madhavi Rao, Neil Spring, Srikanth Sundaresan, Soudeh Ghorbani, Shrikrishna Khare

Agenda

Why DCTCP

Enablement plan

War-story

Takeaways

01 Why DCTCP

DCTCP:

- Detect congestion based on instantaneous queue length
  - Measured by single bit-ECN signal
  - Designed for short-RTT (under ~1ms)
- React before queues overflow in proportion to extent of congestion
- Available in Linux kernel

Refer to Alizadeh, M. et al; Data center TCP; SIGCOMM 10:
https://dl.acm.org/doi/10.1145/1851182.1851192

Safe sharing with DCTCP:

Rack agnostic scheduling

Limited, shallow, shared buffers

Recipe for badness:

Latency / Loss sensitive services badly impacted

DCTCP moderates use of switch buffers – improving network sharing and isolation across services

DCTCP enabled wins

75%

drop in normalized retransmits

38%

drop in read latency (p90/p99) for data-intensive service

DCTCP disabled experiment

Disabling DCTCP for a single region caused 10% drop in throughput (avg) and 4.5x increase in retransmits

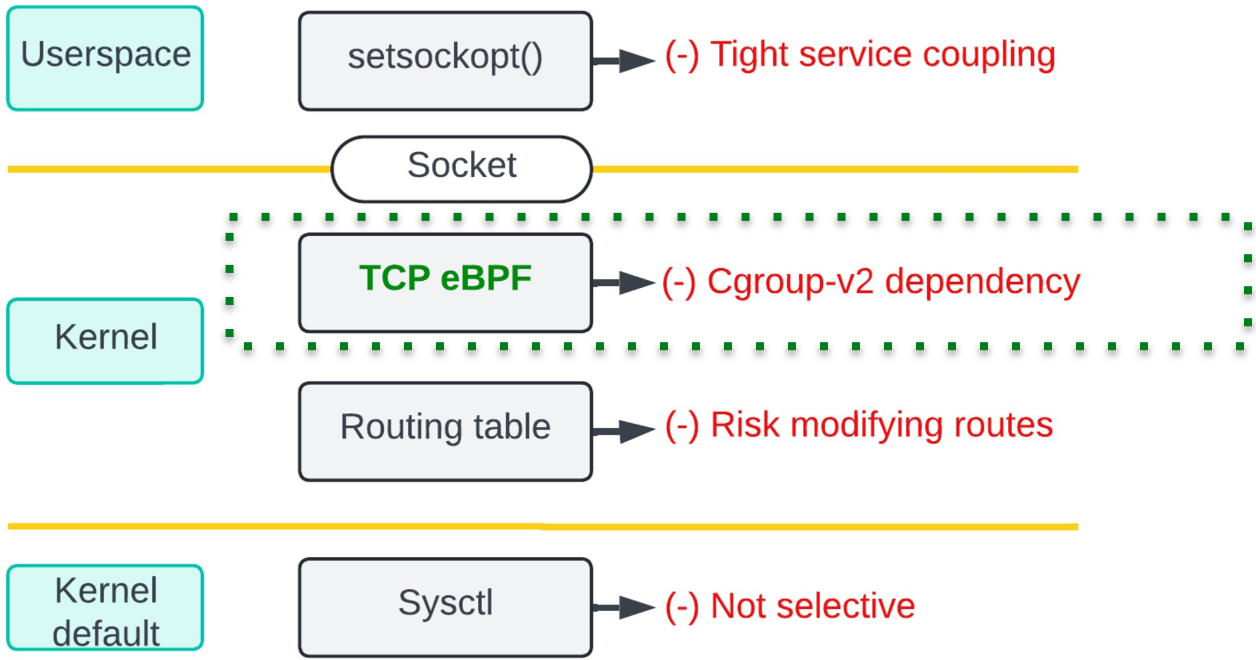
02 Enablement plan

Changing congestion control algorithms at DC/Region scale is hard !

High level goals

	Goal	Reason
1	Target intra-region connections	• Intra-region RTT < 1ms
2	Instantaneous region-wide enablement	• Avoid intra-region DCTCP and Cubic traffic co-existing
3	Minimize configuration complexity	• Cannot work with each service owner
4	Minimize network/service dependency	• Independent from network QoS • Cannot force service restart
5	Fail-open design	• CC tuning failure does not bring down entire network

Available CCA selectors



We created another eBPF based solution for Cgroup-v1

Long lived connections

TCP connections can sometimes be alive for days or longer!

Problem with long-lived connections

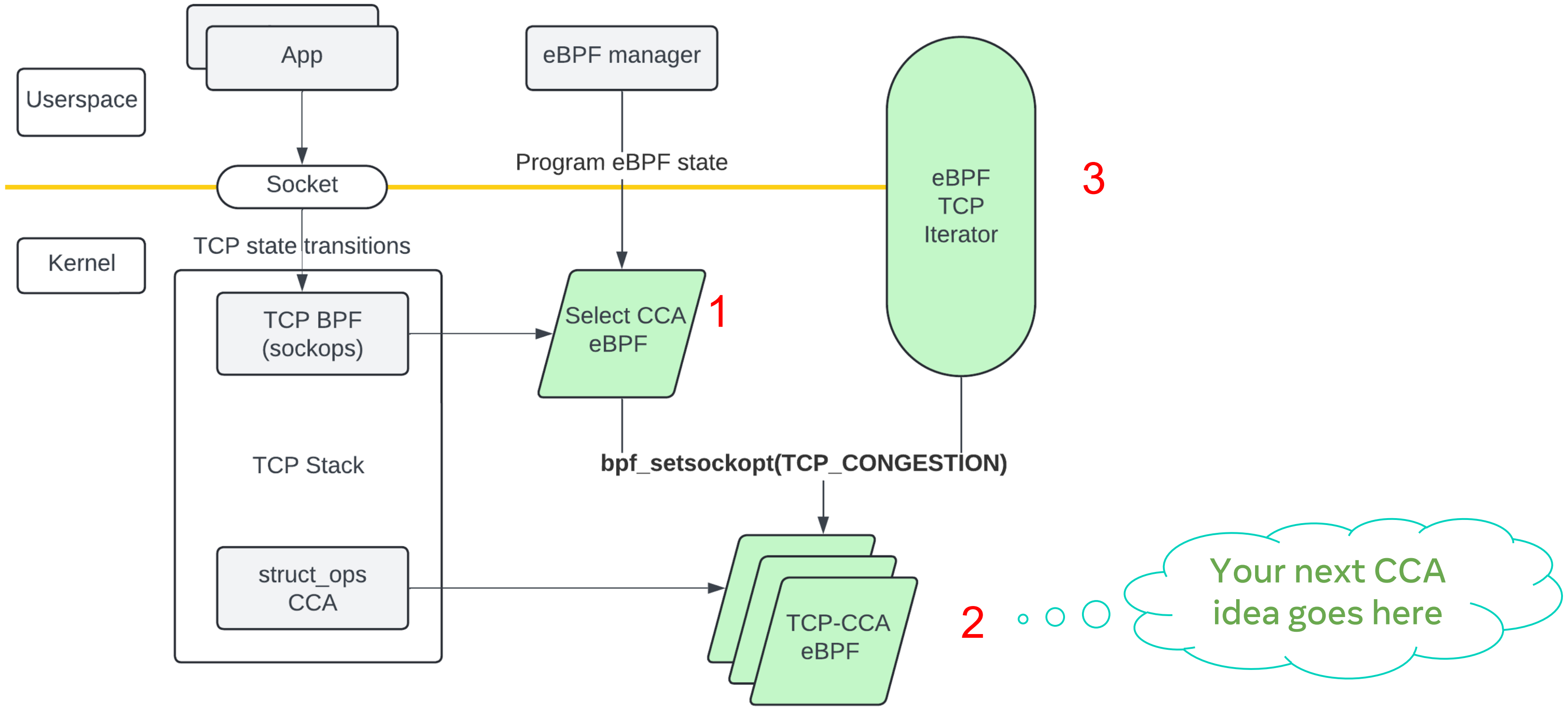
- ECN negotiated at start of the connection (3WHS)
- eBPF knobs triggered during connection establishment

Cannot achieve instantaneous region wide enablement

Solutions:

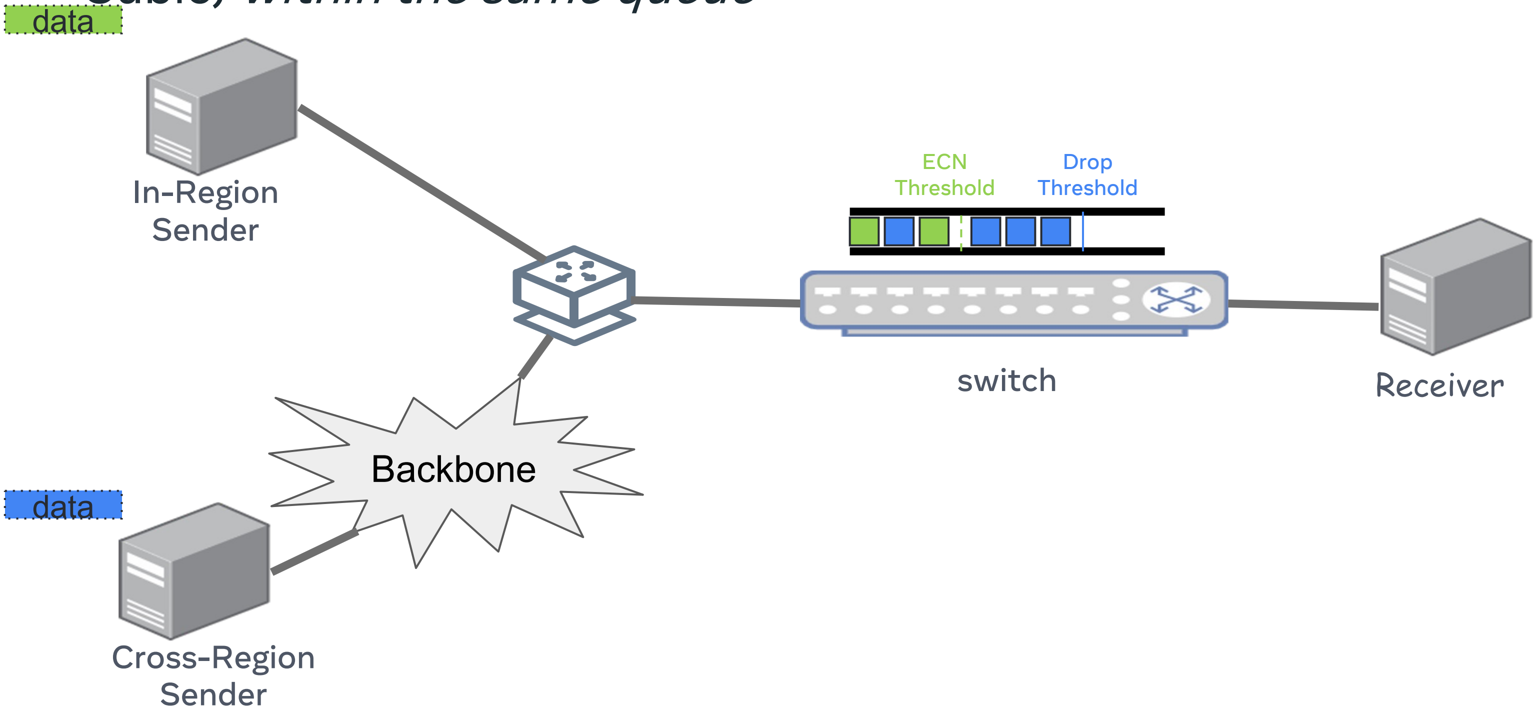
- Disaster recovery framework to drain traffic in phases from a region.
  Each phase can drain 50% traffic from the region
- Built user-triggered eBPF programs (eBPF iterators)

eBPF + CCA:


Userspace
App
eBPF manager
Program eBPF state
Socket
eBPF
TCP
Iterator
3
Kernel
TCP state transitions
TCP BPF
(sockops)
Select CCA
eBPF
1
TCP Stack
bpf_setsockopt(TCP_CONGESTION)
struct_ops
CCA
TCP-CCA
eBPF
2
Your next CCA
idea goes here

Two thresholds per switch queue

- Not enough queues in some ASICs to separate DCTCP and Cubic traffic
  - Balance ECN threshold (for DCTCP) and WRED threshold (for Cubic) *within the same queue*



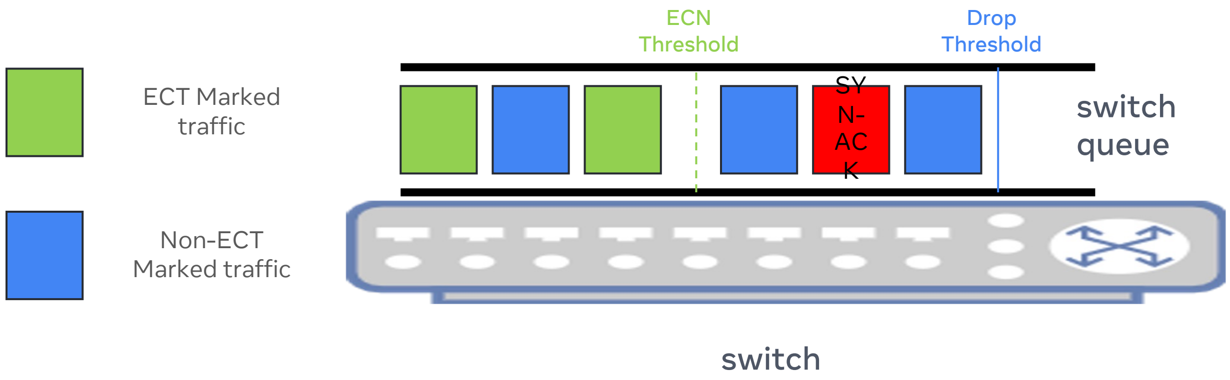
03 War story

Pilot DCTCP rollout and connection timeouts:

- Service saw elevated connection timeouts

- eBPF monitoring could pin-point:
  - Why retransmits (timeout, dup-ack) ?
  - What was retransmitted (SYN/SYN-ACK/TLP/etc) ?

- **Problem:** *Host did not mark SYN-ACK as ECN capable.*
  - **Hack:** SYN-ACK ECN marking eBPF
  - **Solution**: Use ECT bits from requester socket



04 Takeaways

Follow-ups after DCTCP rollout:

- DCTCP struggles with *short* and *heavy* incast bursts
  - Receiver based flow control* to the rescue
- Not all hotspots are ECN-friendly
  - End-Host congestion
- Delay congestion signal solves many of ECN's shortcomings

(*) https://atscaleconference.com/videos/tackling-dc-congestion-and-bursts-balasubramanian-madhavan-and-abhishek-dhamija/

Takeaways:

- Deploying a CCA is not a flip of a switch: Deployment planning needs to consider co-operability and transition performance

- Simple and forgiving CCAs are preferred in a complex and heterogeneous datacenter network

- Hotspots may occur in unexpected places. CCAs must have good fallback

Questions

- Did you deploy ECN only on ToR ?
- What sort of monitoring did you need (find useful) ?
- Example for network hotspot occurring in unexpected places ?
- Why did TLP retransmits increase after deploying DCTCP ?

Meta