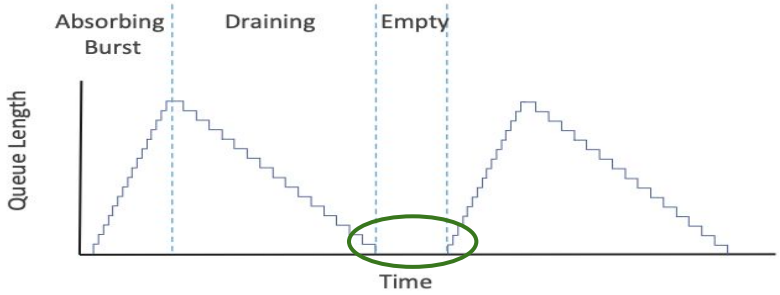
CoDel-ACT:

Realizing CoDel AQM for Programmable Switch ASIC

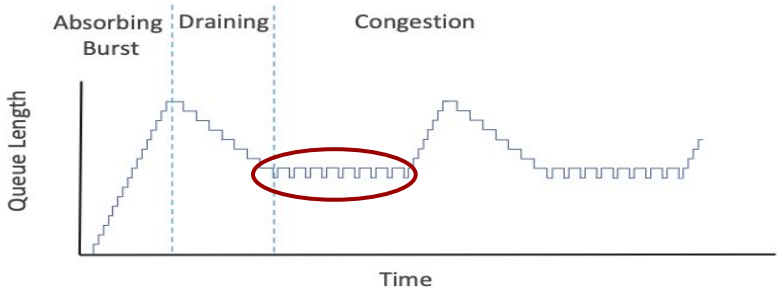
Vedant Bothra, Aditya Peer, Vijay Kumar Singh, Mukulika Maity, Rinku Shah

2024 IFIP/IEEE Networking Conference
June 4, 2024

The Bufferbloat Problem



Ideal queue



Persistently full queue

Bufferbloat problem

Use Active Queue Management (AQM) → RED, CoDel, PIE

Reference: https://tcpcc.systemsapproach.org/aqm.html

State-of-the-art AQM

Use case	State-of-the-art AQM implementations	Flexible	Scalable (support >100s of Gbps)
Last-mile gateways	Software switch implementations [3–7] • Linux kernel, DPDK, bmv2 switch	✔	✗
Backbone networks (Data centers & ISPs)	Fixed function hardware [8–11] • Cisco/Arista switch, Cable modem	✗	✔
	Programmable Network hardware [1–2] • Tofino switch, FPGA-based NIC	✔	✔

Our focus
1. Backbone networks
2. CoDel AQM
   - Parameterless
   - Ease of configuration

References:
[1] R. Kundel et al., “P4-codel: Experiences on programmable data plane hardware,” ICC 2021
[2] A. Sivaraman et al., “No silver bullet: Extending sdn to the data plane,” in Proceedings of the Twelfth ACM Workshop on Hot Topics in
[3] S. Laki et al., “Towards an aqm evaluation testbed with p4 and dpdk, SIGCOMM 2019
[4] P. V¨or¨os et al., “T4p4s: A target-independent compiler for protocol-independent packet processors,” HPSR 2018
[5] C. Papagianni and K. De Schepper, “Pi2 for p4: An active queue management scheme for programmable data planes,” CoNext 2019
[6] G. Ramakrishnan et al., “Fq-pie queue discipline in the linux kernel: Design, implementation and challenges,” LCN 2019
[7] R. Kundel et al., “P4-codel: Active queue management in programmable data planes,” NFV-SDN 2018
[8] https : / / www.cisco.com/c/en/us/products/collateral/switches/catalyst- 9000/white- paper- c11-742388.html
[9] https://www.arista.com/en/um-eos/eos-quality-of-service#xx1166435
[10] https://www-res.cablelabs.com/wp-content/uploads/2019/02/28094021/DOCSIS-AQMMay2014.pdf
[11] T. Høiland-Jørgensen, D. T¨aht, and J. Morton, “Piece of cake: a comprehensive queue management solution for home gateways,” LANMAN 2018
[12] X. Du, K. Xu, L. Xu, K. Zheng, M. Shen, B. Wu, and T. Li, “R-aqm: Reverse ack active queue management in multitenant data centers,” ToN 2022

Controlled Delay (CoDel)

```
define INTERVAL 100 ms
define TARGET 5 ms

delta = count - lastCount
lastCount = count

if (delta > 1) and (now - dropNext < 16*INTERVAL):
    count = delta
else:
    count = 1
dropNext = now + INTERVAL/sqrt(count)
```

codel_init() function

codel_init() function depends on historical count values

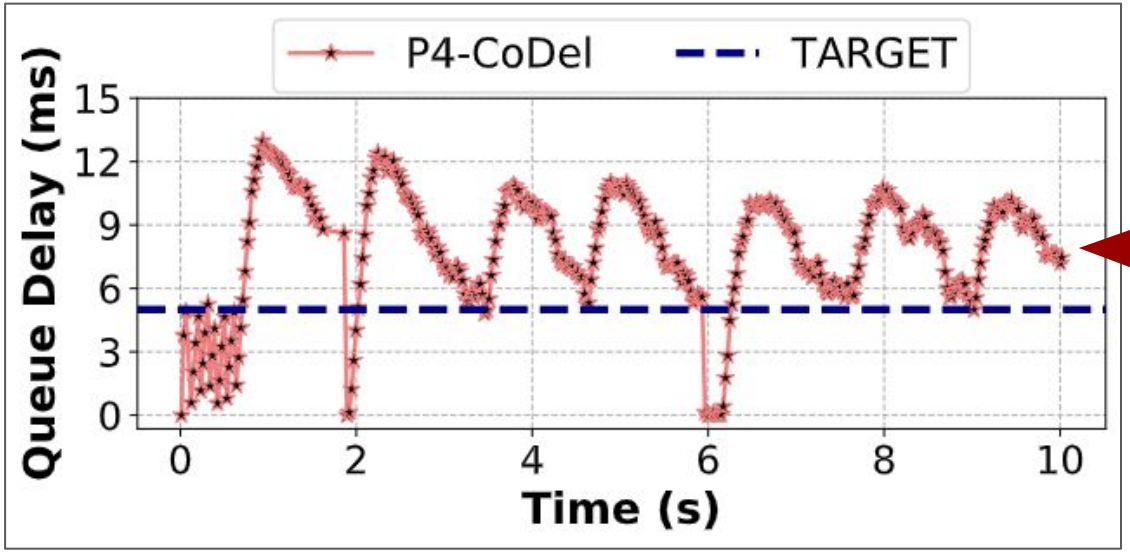
Recovering from congestion?
- NO: count = historical value
- YES: count = 1

```
# Here, dropping = True.
# Sender should have responded to packet drop
    by now
if (now >= dropNext):
    # Drop the packet
    count = count + 1    # Update "count"
    # Calculate drop next time
    dropNext += INTERVAL/sqrt(count)
```

codel_update() function

if **queue_delay > TARGET?**
- **DROP** the first packet
- **UPDATE** dropNext time
  - dropNext: wait time before dropping next packet
- Until dropNext time
  - Increment count for subsequent packets

Does existing programmable switch-based CoDel design effectively solve Bufferbloat?



Queue delay > TARGET

P4-CoDel[1] : Queue delay for 10 parallel TCP flows

Average Queue delay = 8.3 ms

Does not maintain historical state!

References:
[1] R. Kundel et al., “P4-codel: Experiences on programmable data plane hardware,” in ICC 2021.

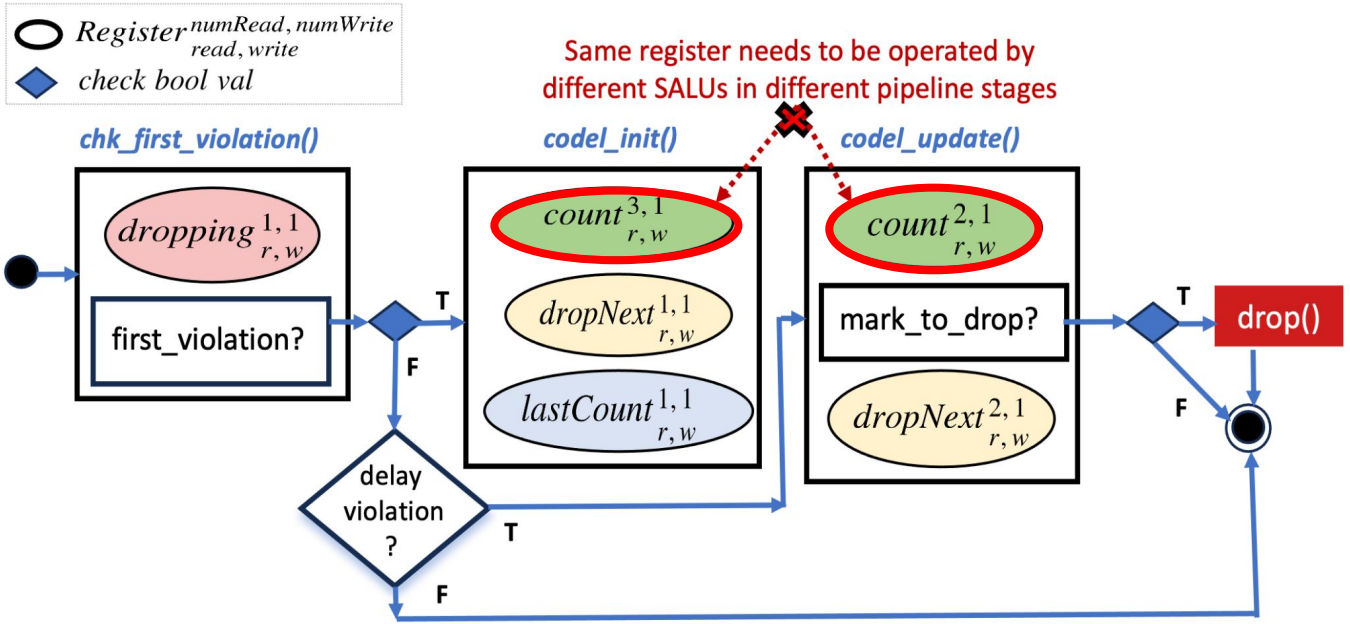
CoDel-ACT's Key Idea

(Re)Design CoDel AQM

- Adapt packet drop rate
  - based on **historical packet drop count**
  - RFC-compliant
- Operates at line rates
  - Runs **entirely in the data plane**
- Amenable to be implemented on Intel Tofino switch

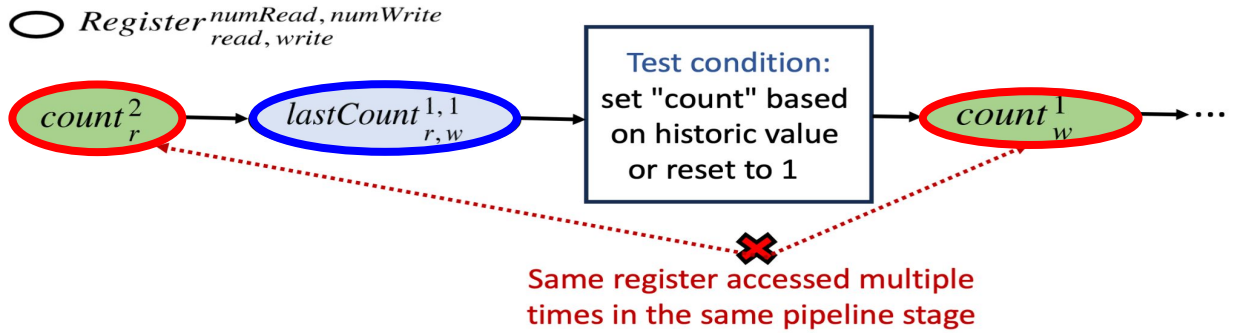
Design challenge I

"Same register cannot be accessed across different switch pipeline stages"



Design challenge II

"A packet can access a single register only once (either read/write/ RegisterAction)"

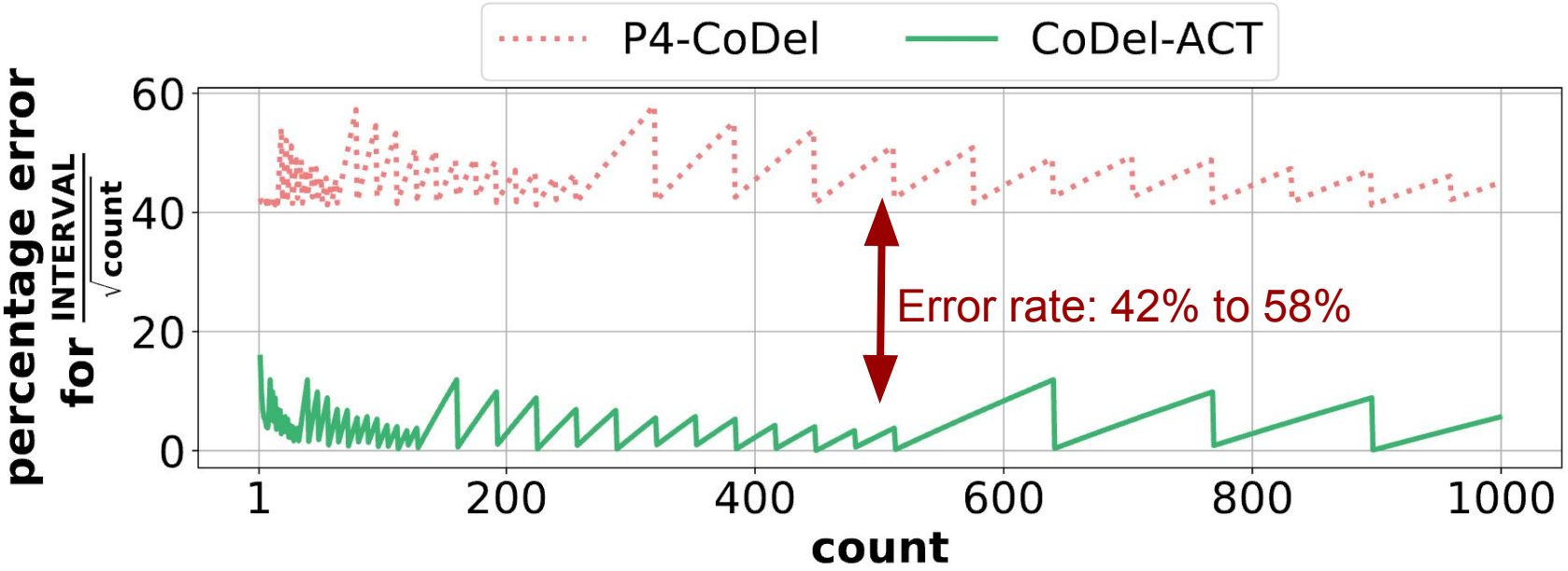



```
define INTERVAL 100 ms
define TARGET 5 ms

delta = count - lastCount
lastCount = count

if (delta > 1) and (now - dropNext < 16*INTERVAL):
    count = delta
else:
    count = 1
dropNext = now + INTERVAL/sqrt(count)
```

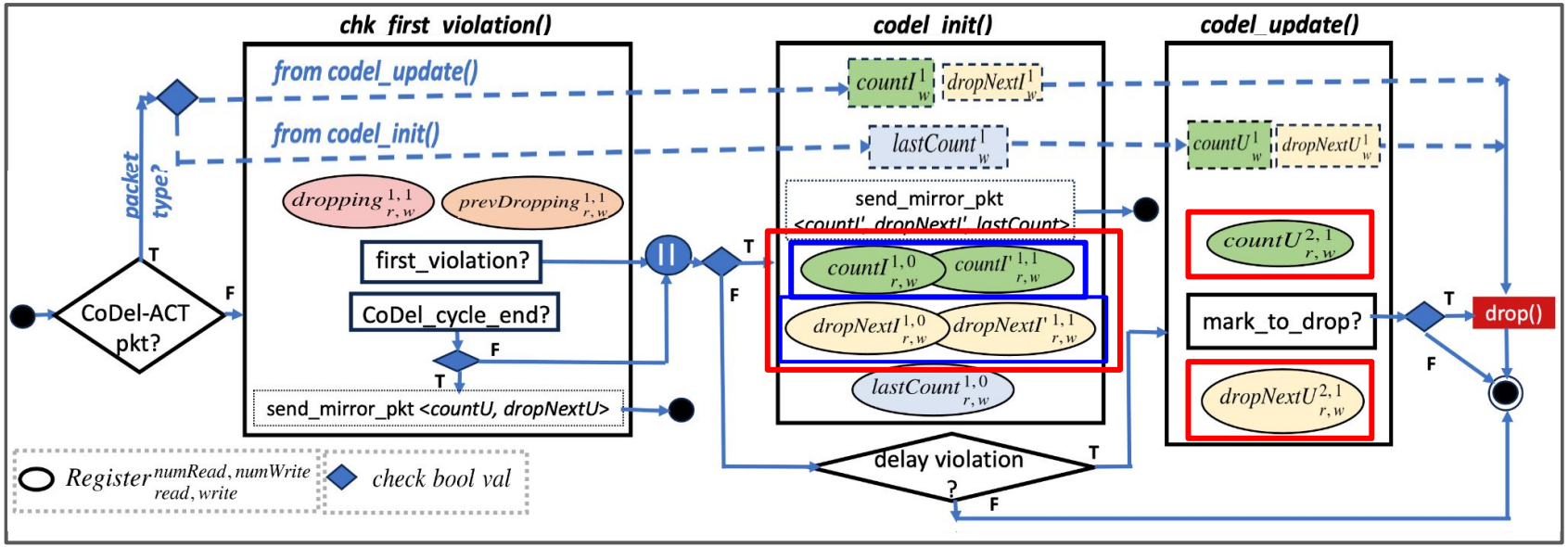
codel_init() function

Design challenge III

“High error rate for dropNext computation.”



CoDel-ACT design



Shadow Registers
- **Within single stage :**
  - count
  - dropNext

Shadow registers
- **Across stages:**
  - count
  - dropNext

Register State Synchronization
- **First delay violation**
  - Sync after codel_init()
- **Congestion cycle ends**
  - Sync after codel_update()

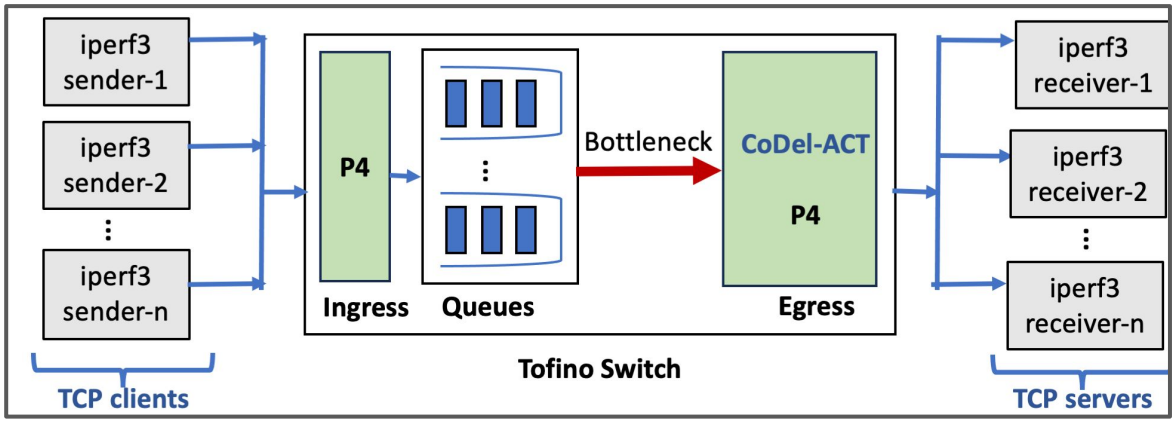
Evaluation Questions

1. How does CoDel-ACT **perform** compared to state-of-the-art?

2. How **aggressive** is CoDel-ACT compared to state-of-the-art?

3. What is the impact of packet recirculation on **switch resource utilization**?

Experiment setup



CoDel parameters:

- TARGET = 5 ms
- INTERVAL = 100 ms

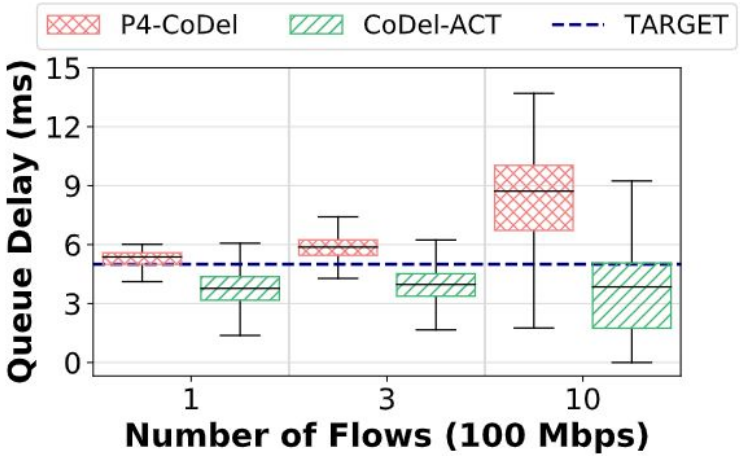
Setup:

- AMD Ryzen 9 5950X
- Aurora 610 Intel Tofino switch
- Congestion emulated by rate limit on Tofino'sTM

Workload/Tofino configuration:

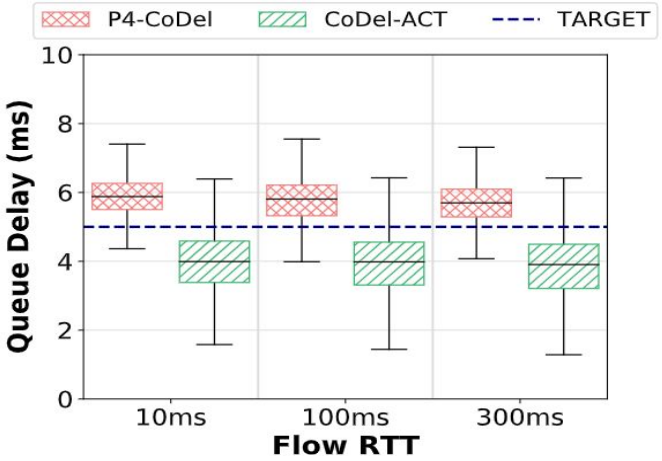
- Parallel TCP flows using "iperf3"
- Emulated flow RTT using "tc"
- Total packet rate = 90% of bottleneck bandwidth

CoDel-ACT vs. P4-CoDel performance



Varying: Number of Flows

Bottleneck bandwidth = 100 Mbps



Varying: Flow RTT

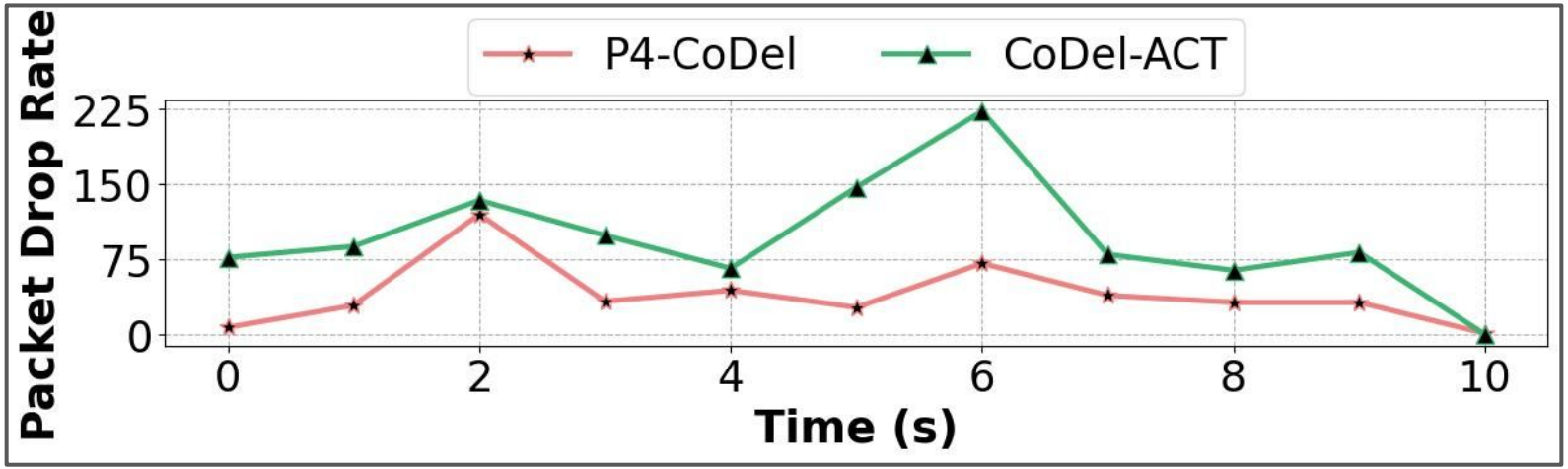
Parallel TCP flows = 10

Bottleneck bandwidth = 100 Mbps

Varying number of flows:
Average queue delay
- CoDel-ACT < TARGET
- P4-Codel exceeds TARGET
  - Up to 43 %.

Varying RTT:
Average queue delay
- CoDel-ACT < TARGET
- P4-CoDel > TARGET

How aggressive is CoDel-ACT compared to P4-CoDel?



Number of parallel TCP flows = 10
Bottleneck bandwidth = 100Mbps

CoDel-ACT drops more packets
=> more aggressive
=> Quick congestion recovery

Conclusion

- Implemented RFC-compliant CoDel on Intel Tofino switch
- Compared to state-of-the-art
    - Average queue delay (⬇️52%)
    - Worst-case bandwidth wastage (4%)

Future work

- Reduce state synchronization delays