Detecting Stragglers in Programmable Data Plane

20/08/2025

Riz Maulana, Habib Mostafaei, Nirvana Meratnia

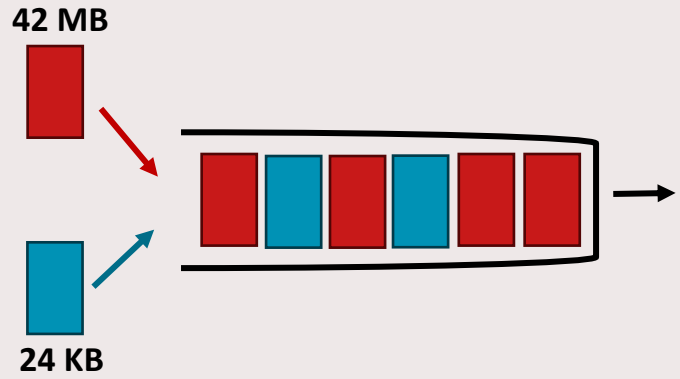
IRIS Cluster, Department of Mathematics & Computer Science

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Flow scheduler

Goal: minimize flow completion time (FCT).

- Prioritizing small flows over larger flows.



Types of flow scheduler

Clairvoyant

- Requires detailed flow information
- pFabric [1], pHost [2], Homa [3]

Non-clairvoyant

- Operates with limited flow information
- PIAS [4], Qclimb [5]

What can go wrong?

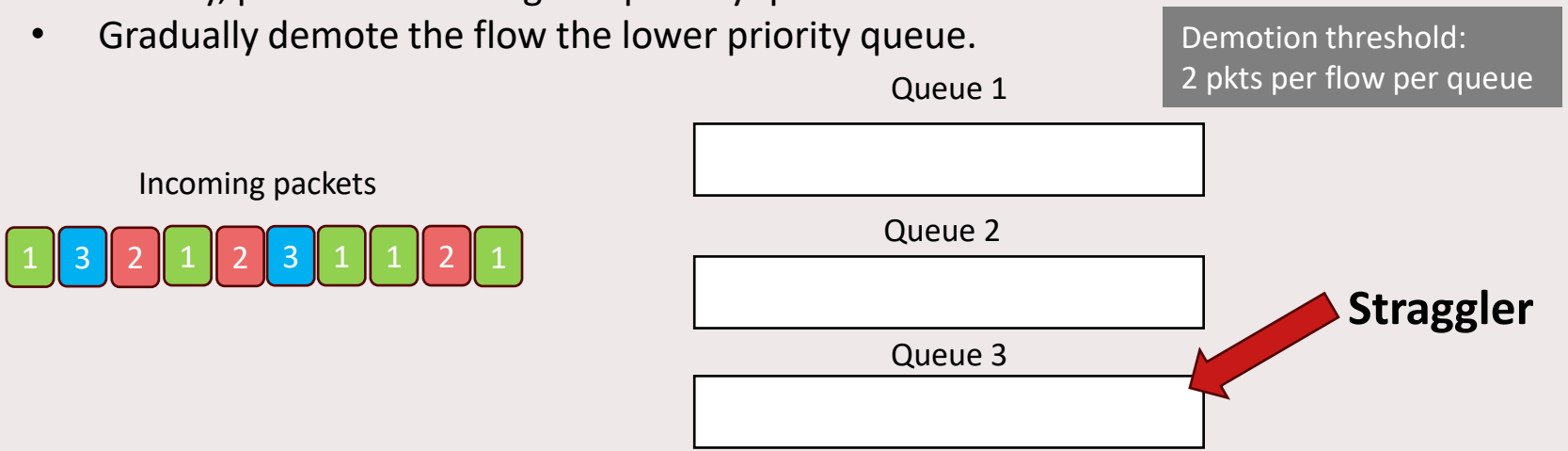
- Prioritizing certain flows -> push others into lower priority queues.
  - Accumulate longer waiting times.

- **Straggler**: a packet with *unexpectedly higher queueing delay* compared to other packets from the same flow.

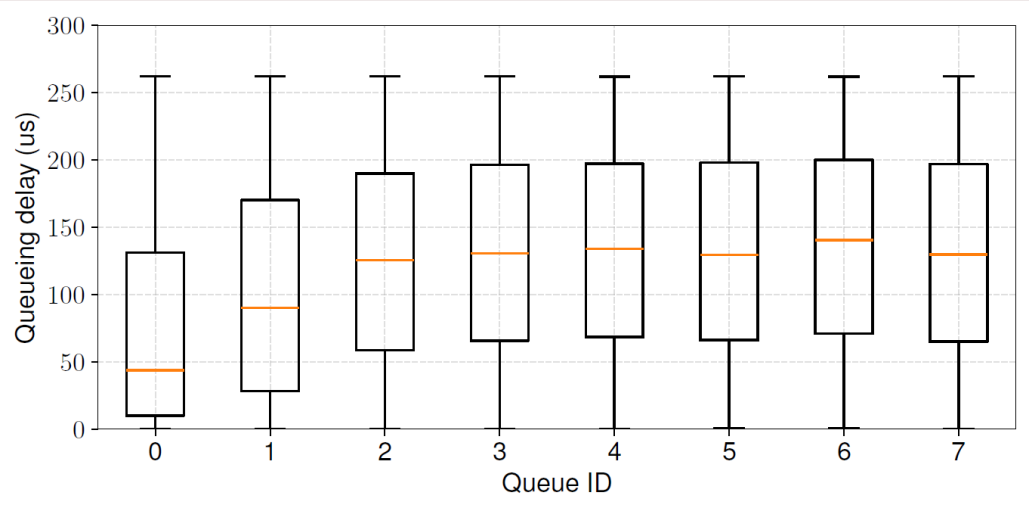
- Straggler is well-studied in system community.
  - Tasks that run much slower than other tasks in the same job.

Stragglers in network flows

- Example: PIAS Scheduling.
  - Basic idea: Prioritize small flows.
  - Initially, put the flow in highest priority queue.
  - Gradually demote the flow the lower priority queue.



Queueing delay distribution



Measure queueing delay in hardware switch.

- PIAS 8 queues
- CAIDA traces (2016).

- High priority queues -> lower queueing delay.
- Demotion to lower priority queue -> higher queueing delay.

TU/e

The need for straggler detection

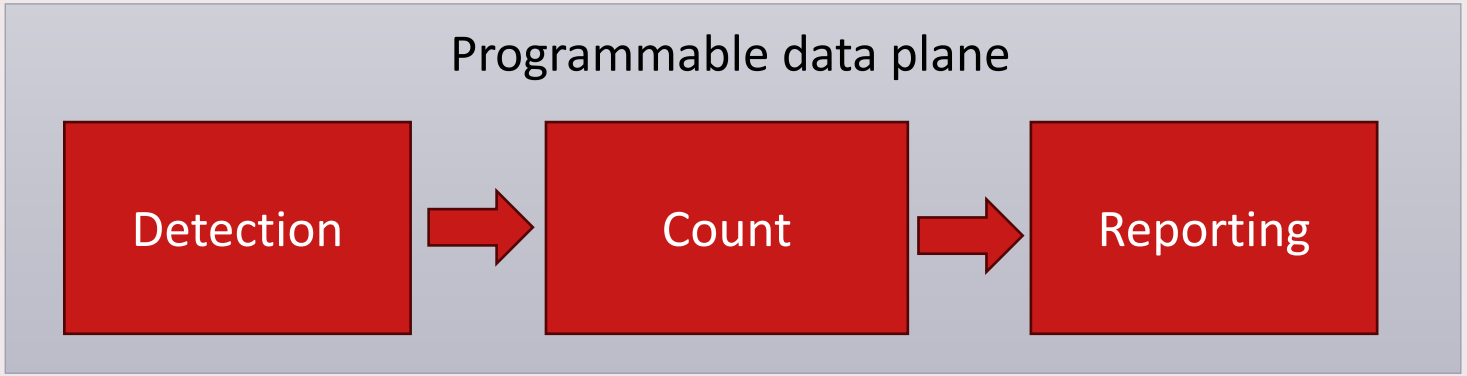
- Provide insights into flow scheduler behavior.
- Diagnose or troubleshoot the flow scheduler.
- Leverage granular visibility of programmable data plane.

Contribution

- Propose **StragFlow**, a tool for detecting stragglers.

- A new perspective in queue monitoring.
  - Previous works: identifying the ***root cause*** of the issue.
    - ConQuest [6]: which flows occupy the most space in a queue?
    - PrintQueue [7]: which flows (directly/indirectly) cause high queueing delay?
  - **StragFlow**: identifying the ***victims*** of the issue.

StragFlow components



*** Limitation of programmable data plane:**

- No floating point
- Tens of MB memory
- Pipelined architecture
- Limited number of stages

StragFlow: detection

Threshold

- Set a static threshold to detect stragglers

Moving average

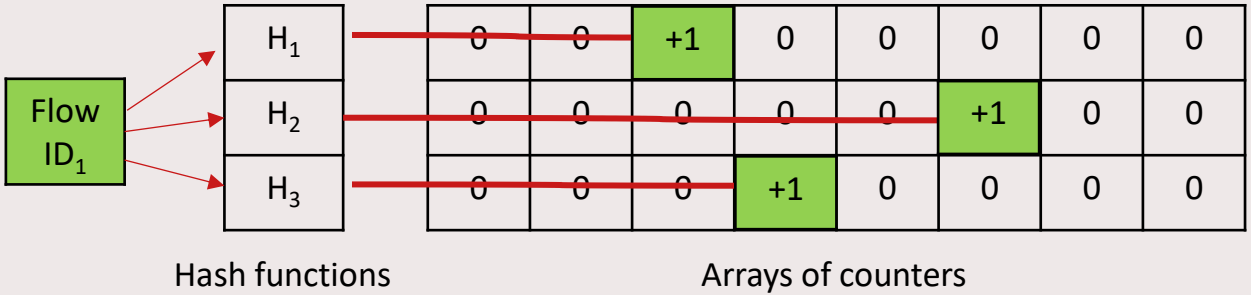
- Flexible threshold
- A gradual change in latency may slowly drift the average
  - Eventually making it less sensitive to emerging stragglers

Compare with previous (highest) queueing delay

- Keep track of the delay history
- No global threshold

StragFlow: counting stragglers

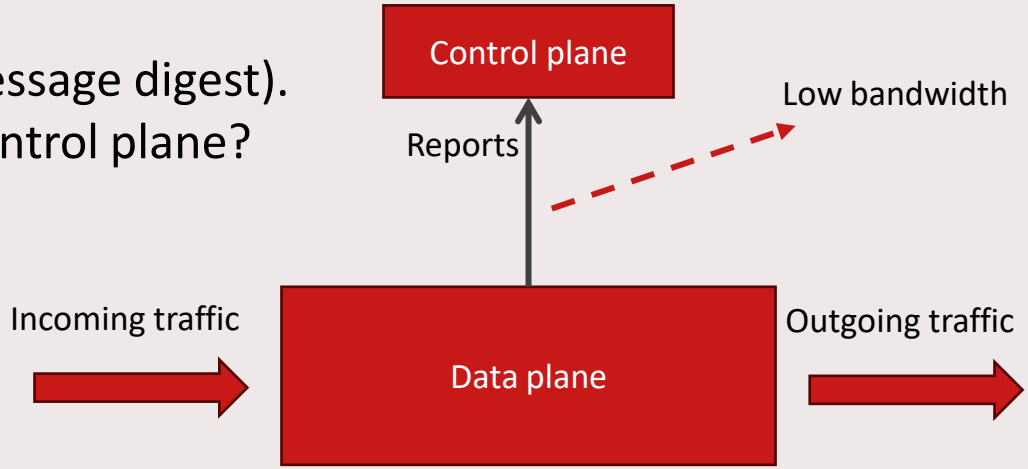
- Use a compact, collision-tolerant data structure.
  - Count-min sketch (CMS)
- Insertion (current queueing delay > previous (highest) queueing delay)



StragFlow: reporting stragglers

- Naïve approach: reports all stragglers.
  - Will flood the control plane.

- Send only necessary info (message digest).
- How to avoid flooding the control plane?

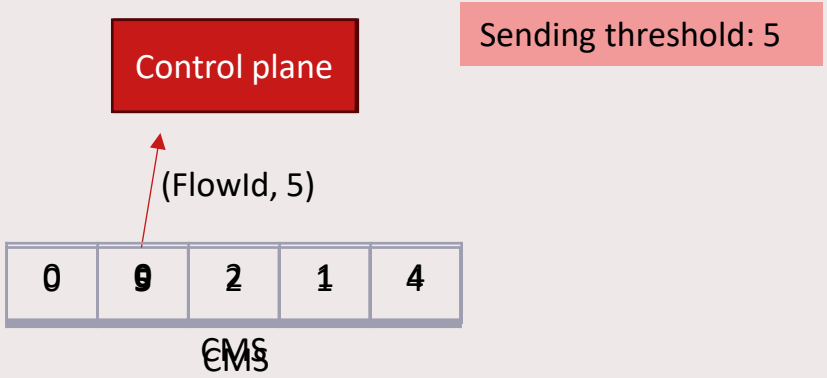


Reducing report rates

Accumulates the count until ***sending threshold.***

When reaching the sending threshold:

- Send the count to control plane
- Reset the count to 0.



Sending threshold: 5

Small sending threshold

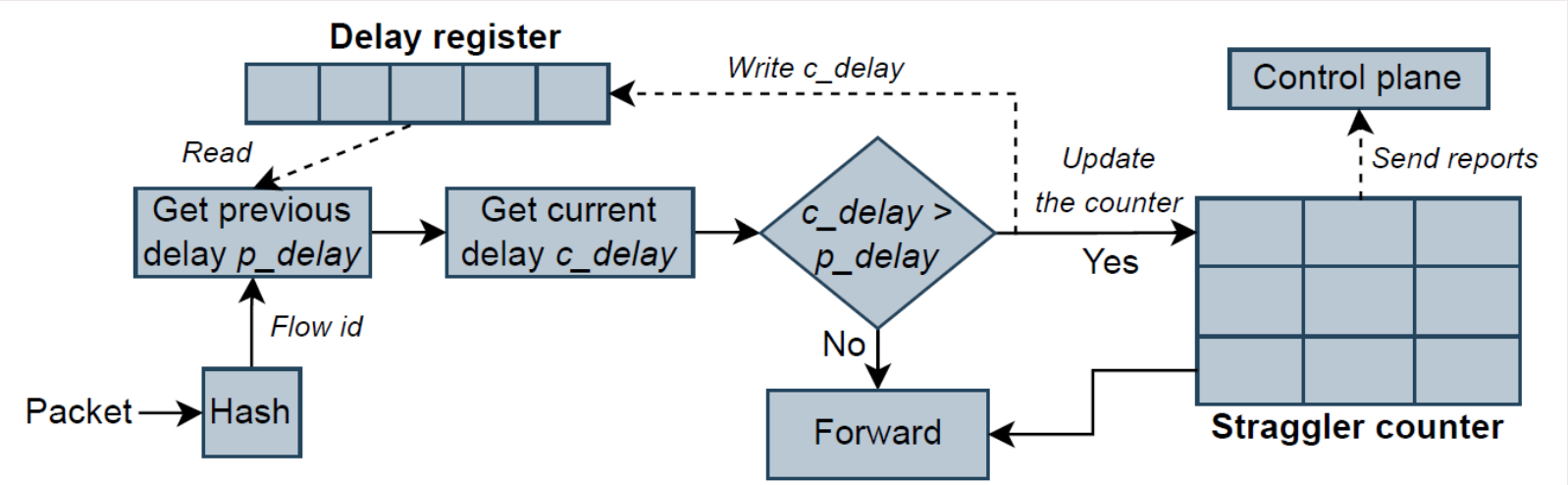
Pros:

- Detect stragglers in small flows
- Many small counter
- Lower probability of hash collision

Cons:

- Might miss some stragglers

StragFlow: Overall detection mechanism

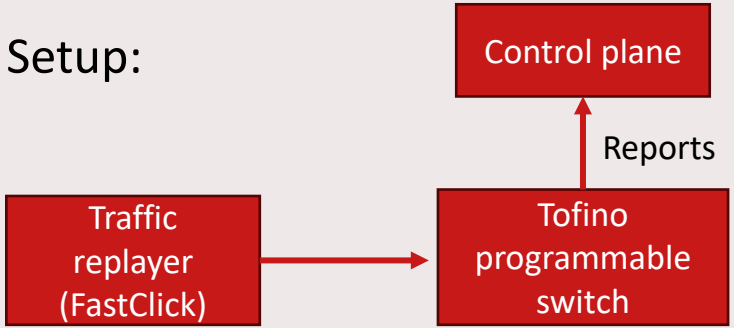


Evaluation

Goal:

- Detect stragglers when using different scheduling mechanisms.
- 8-queue PIAS vs 1-queue FIFO.

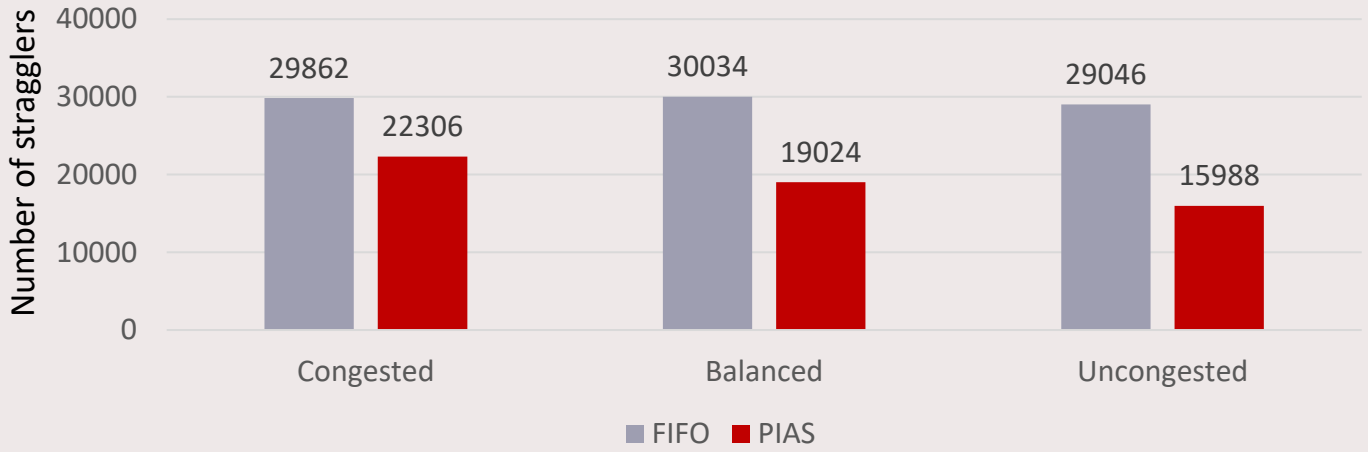
Setup:



Dataset: CAIDA network traces (> 30M packets)

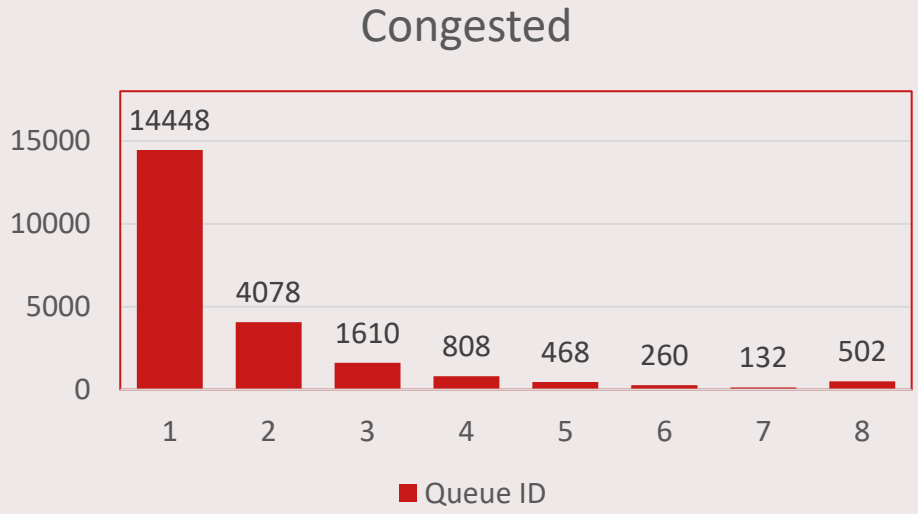
Scenario	Ingress bandwidth	Egress bandwidth
Congested	1 Gbps	0.8 Gbps
Balanced	1 Gbps	1 Gbps
Uncongested	1 Gbps	1.2 Gbps

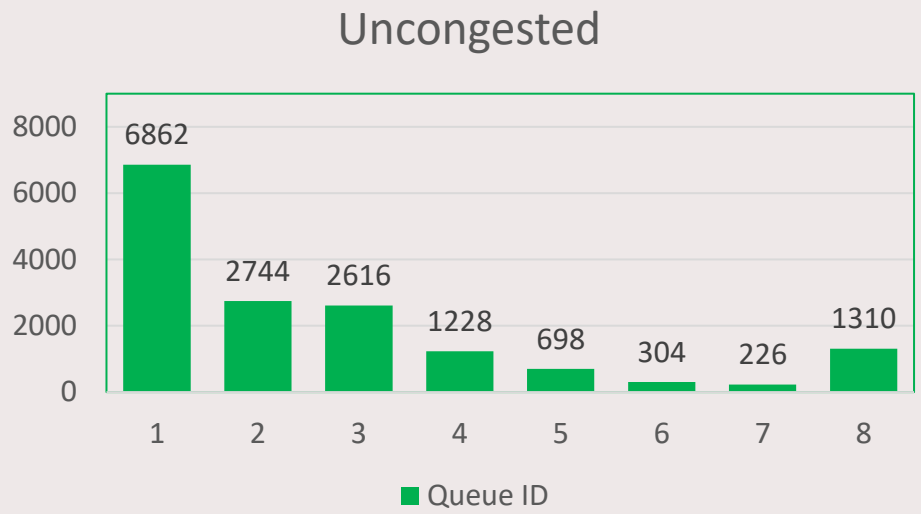
Evaluation: straggler detection



- PIAS has fewer stragglers than FIFO
  - Prioritizing small flows mitigates queueing delays more effectively than a FIFO queue.

Evaluation: traffic pattern on queue-level (8-queue PIAS)





- Congested queues might indicate misconfiguration or need readjustment.

Evaluation: resource consumption

Resource type	Resource consumption*
Match crossbars	7.29%
Gateway	8.33%
Hash bits	7.53%
SRAM	12.92%
TCAM	0%
Stateful ALU	0%
Number of stages	6 out of 12

*Including other functions such as forwarding and PIAS scheduling.

StragFlow fits comfortably within the programmable data plane.

Conclusion

- Flow schedulers can cause stragglers.

- Proposed **StragFlow**: a tool for detecting stragglers.
  - Instead of identifying the root cause, **StragFlow** shifts the focus to detecting the victims.
  - Operates efficiently in programmable switches.
  - Effectively identifies stragglers and reveals insights into how scheduling policies impact flow performance.

Thank you

Reference

[1] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, “pfabric: minimal near-optimal datacenter transport,” ser. SIGCOMM ’13, 2013.
[2] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, “phost: distributed near-optimal datacenter transport over commodity network fabric,” ser. CoNEXT ’15, 2015.
[3] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, “Homa: a receiverdriven low-latency transport protocol using network priorities,” ser. SIGCOMM ’18, 2018, p. 221–235.
[4] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, “Information-Agnostic flow scheduling for commodity data centers,” in NSDI ’15, Oakland, CA, May 2015, pp. 455–468.
[5] W. Li, X. He, Y. Liu, K. Li, K. Chen, Z. Ge, Z. Guan, H. Qi, S. Zhang, and G. Liu, “Flow scheduling with imprecise knowledge,” in NSDI ’24, 2024.
[6] X. Chen, S. L. Feibish, Y. Koral, J. Rexford, O. Rottenstreich, S. A. Monetti, and T.-Y. Wang, “Fine-grained queue measurement in the data plane,” in CoNEXT ’19, 2019, p. 15–29.
[7] Y. Lei, L. Yu, V. Liu, and M. Xu, “Printqueue: Performance diagnosis via queue measurement in the data plane,” ser. SIGCOMM ’22, 2022, p. 516–529.