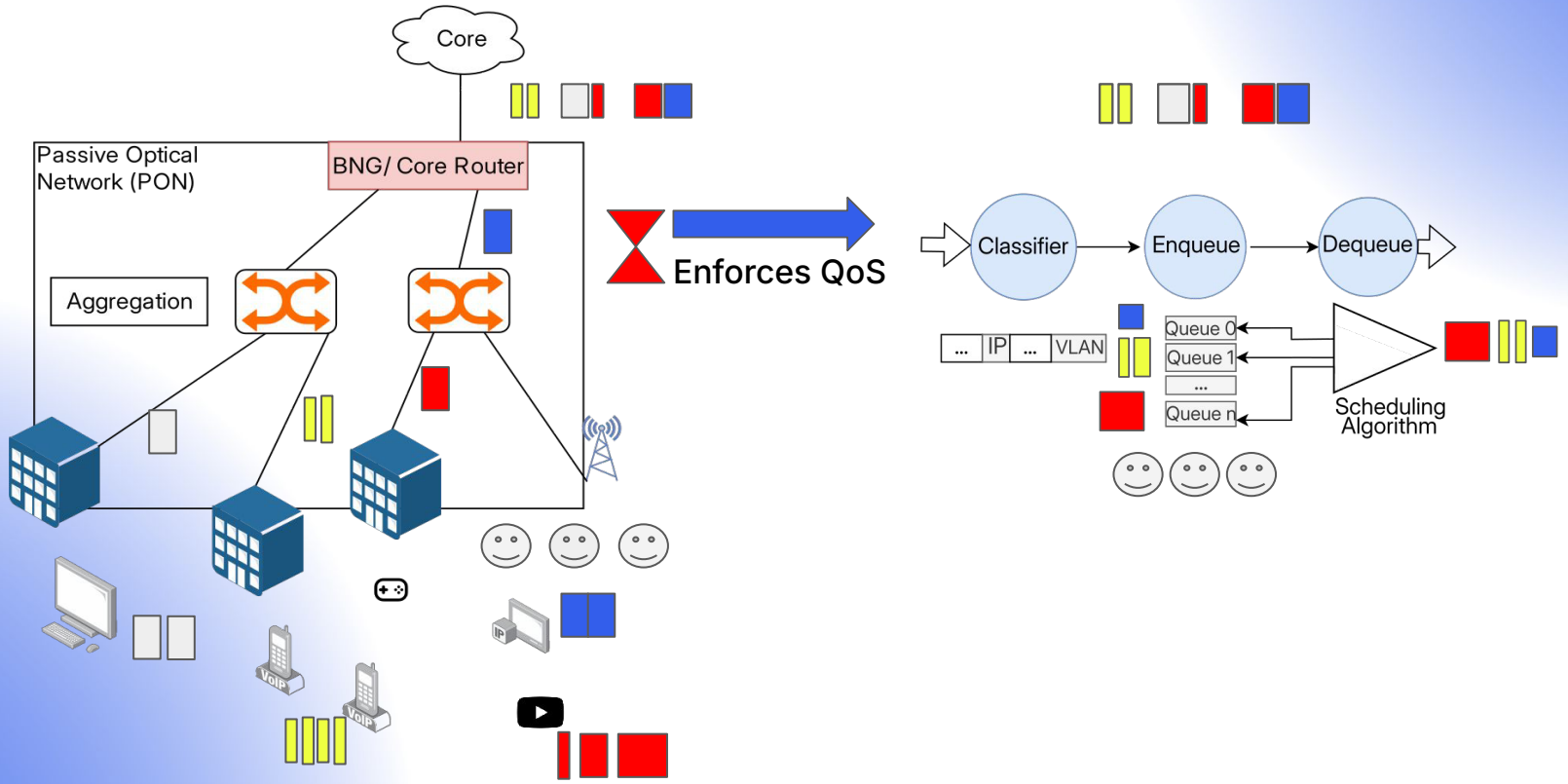
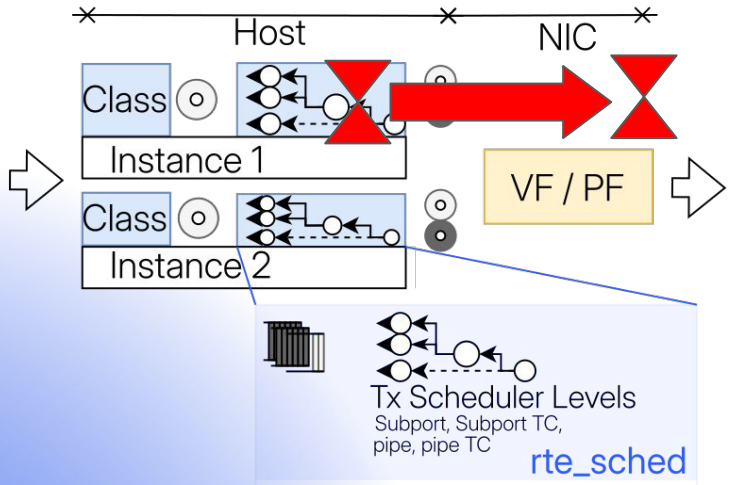
Bridging DPDK and NIC HQoS

With Priority-Aware Backpressure (PAB)

Rubens Figueiredo
Hagen Woesner
Andreas Kassler
Holger Karl





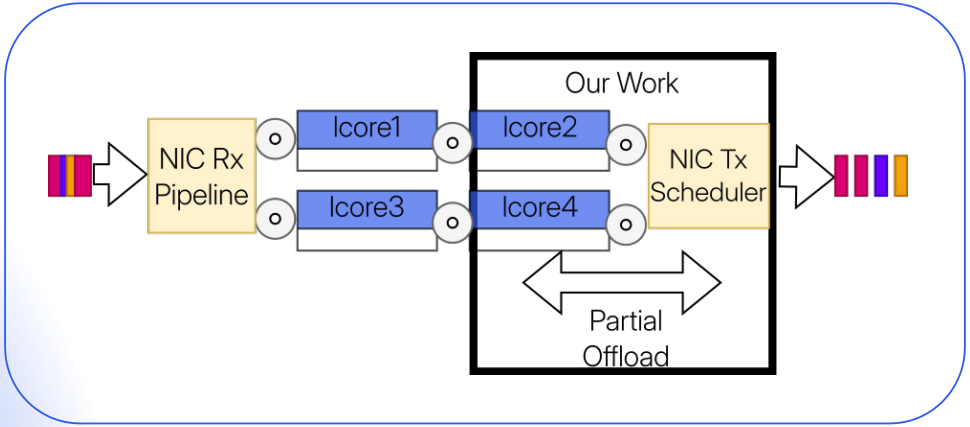
Traffic shaper scalability
Adding more instances, VFs

Performance Limitations
Lock contention during global rate limiting [1]
Scheduler/ shaper coordination

[1] Köppeler, Jonas, Toke Høiland-Jørgensen, and Stefan Schmid. "Have your CAKE and eat it too: Scaling software rate limiting across CPU cores." *2025 IEEE 31st International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2025.

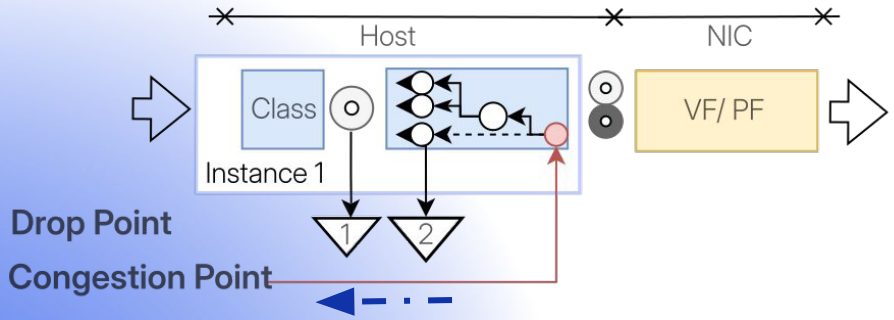
Offloading Shapers

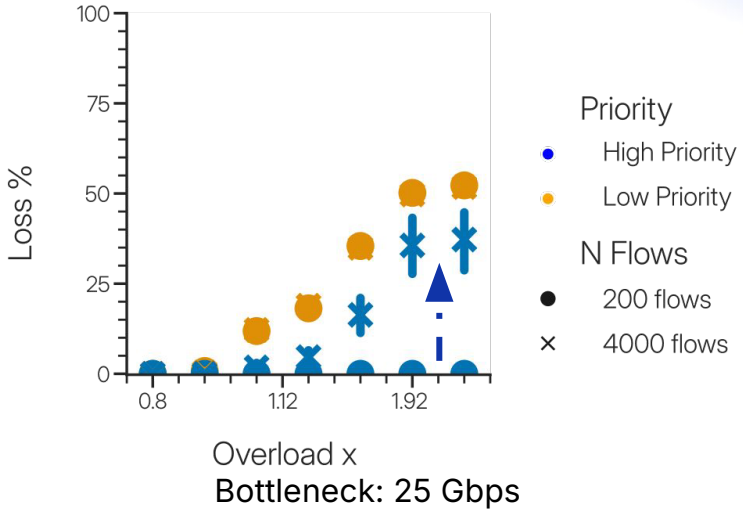
Hybrid Software-Hardware QoS Data Plane



Scheduler Overload

Single Instance Loss Behavior

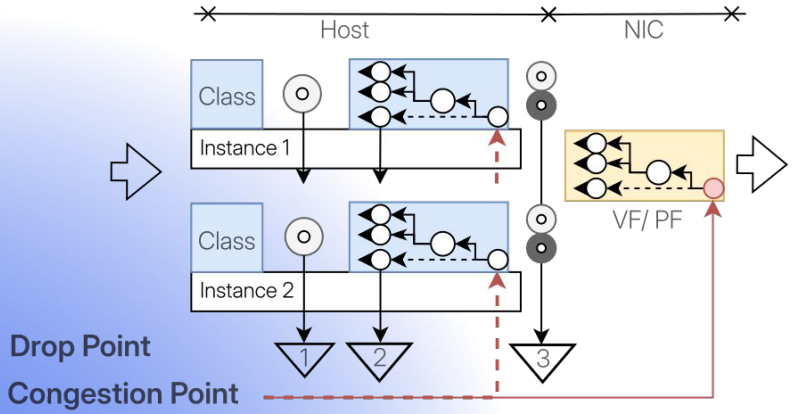


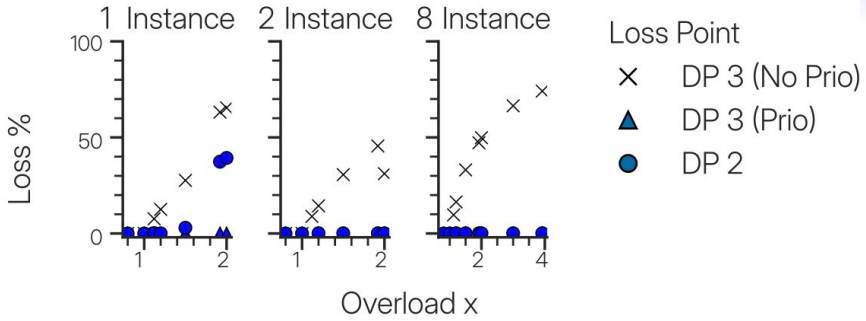


Bottleneck: 25 Gbps

Scheduler Overload

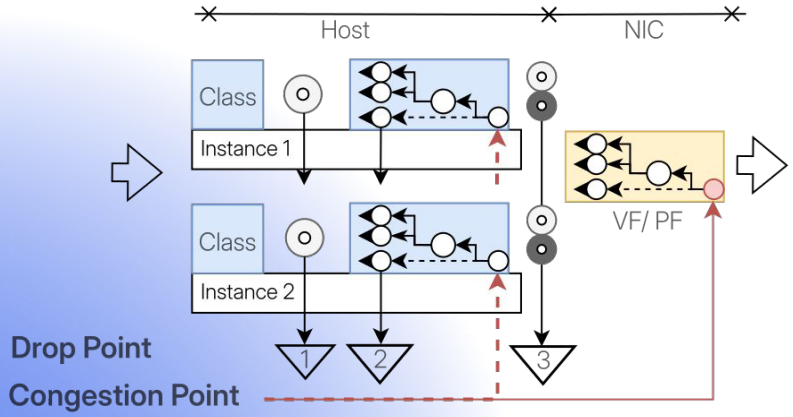
Multiple Instance Loss Behavior

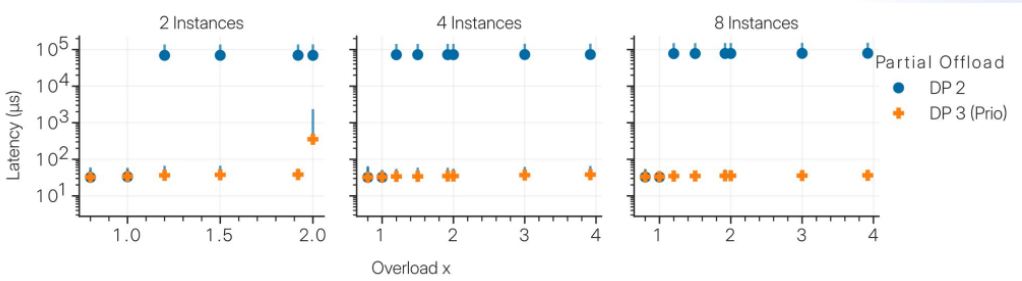




Scheduler Overload

Multiple Instance Loss Behavior





Scheduler Partial Offload

Design Goals

Software

- Flexible deployment
- Flow/ CPU scalability
- Fine-grained scheduler and shapers

Hardware

- Aggregate traffic shaping
- Commodity high-speed hardware
- RTE_TM support

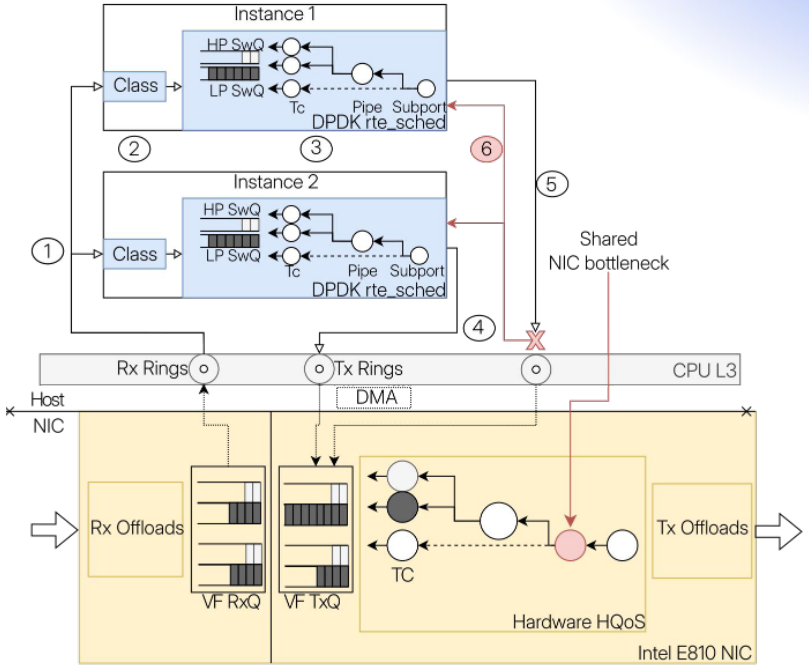
Coordination Gap

- No NIC backpressure
- **QoS priority/ fairness violations**

Dequeue Control

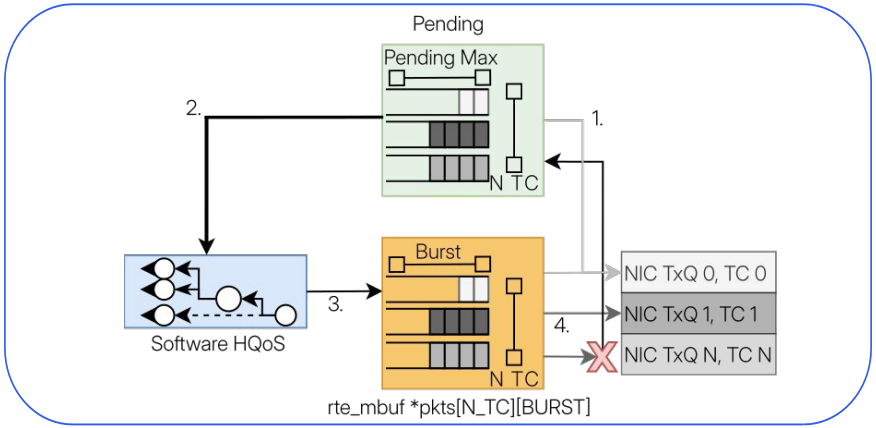
Using NIC Backpressure

1. Pkt Receive
2. Classification
3. Rte_sched Enqueue/ Dequeue
4. Successful Tx_burst
5. **Failed Tx burst**
6. **Packets returned to user space**
7. **Apply Backpressure**



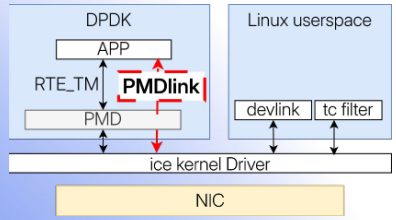
Priority-Aware Backpressure (PAB)

1. Retry Pending Packets
2. Compute Dequeue Allowance
3. Dequeue New Packets
4. Transmit New Packets

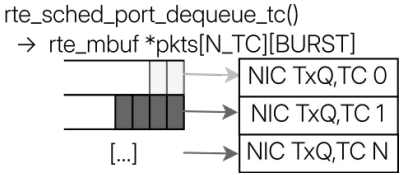


PAB Main Mechanisms

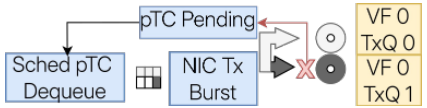
Configuration API



Sched TC Dequeue



Dequeue Adjustment

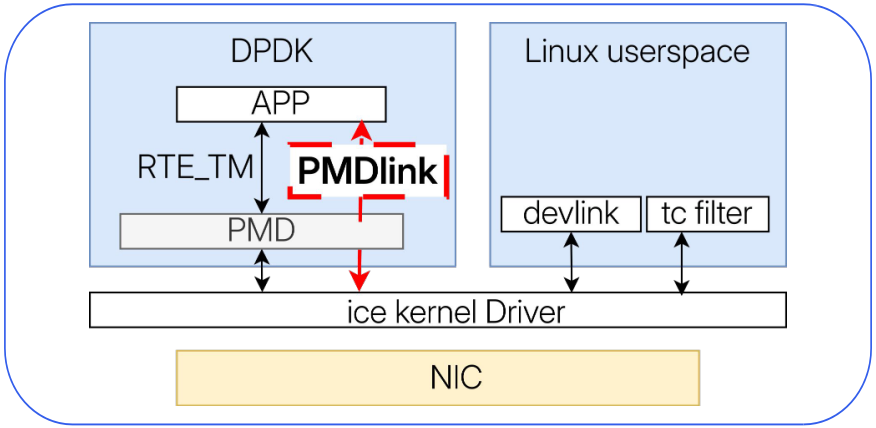


E810 Switchdev

DPDK + Linux

Switchdev/ devlink configures
- Forwarding rules
- QoS parameters

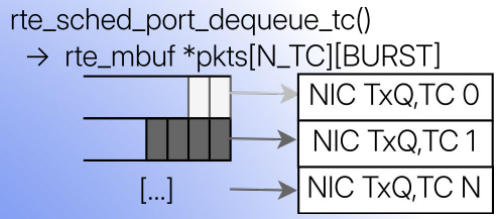
DPDK manages the VFs



Per-TC Dequeue

Changing rte_sched Return

Scheduler returns packets grouped by traffic class




```
int
rte_sched_port_dequeue_tc(
        struct rte_sched_port *port,
        struct rte_mbuf ***pkts,
        uint32_t n_pkts,
        uint32_t *burst_allowed,
        uint32_t *tc_counts)
```

- Return Structure
- Total Burst
- pTC Allowed # pkts
- pTC # Dequeued pkts

PAB Mechanism

Per-TC Dequeue Allowance

1. Burst_allowed based on Pending length
2. Scheduler is limited by burst_allowed

1

```
uint16_t space = PENDING_MAX - pending[tc].cnt;
burst_allowed[tc] = RTE_MIN(space, burst_conf.qos_dequeue);
```

2

```
grinder_schedule(..., uint32_t *burst_allowed) {
    if (burst_allowed[tc]  == 0) {
        return 0;
    }
    ...
    port→pkts_out_tc[tc][port→tc_pkts_out[tc]++] = pkt;
    ...
    if (burst_allowed) {
        burst_allowed[tc]--;
    }
}
```

PAB Evaluation

Evaluation Criteria

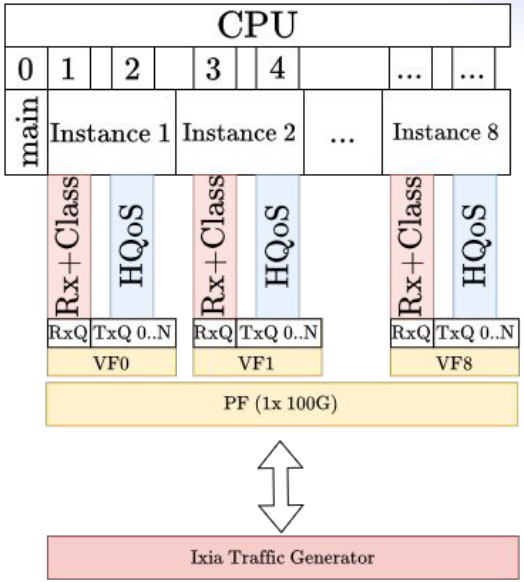
N Flows 200-32k (pipes)
TC (1-8)
Instances (1-8)

Metrics

Throughput (Gbps)
Latency (Min, Max, Avg, ECDF)
Loss %

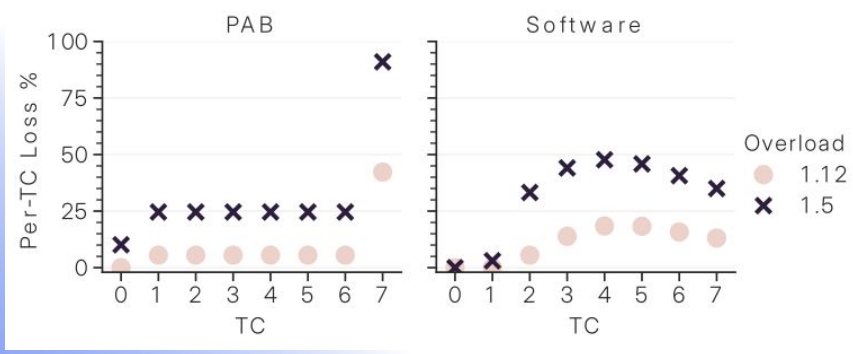
Offered Load

2-9 Mpps
25-98 Gbps



Single Instance PAB

Loss Distribution Across TCs

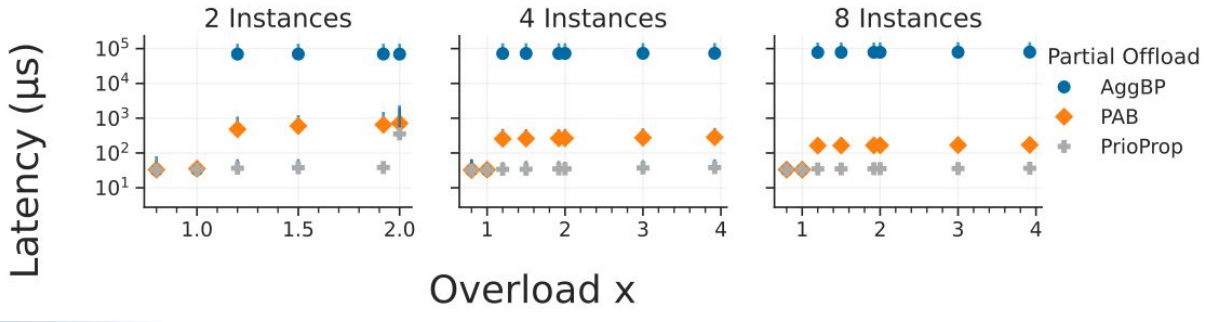


N TC	Mode	Latency (µs) Min	Avg	Max
2	PAB	33.25	93.58	370.09
	Software	32.56	261.56	3188.26
4	PAB	33.34	310.95	817.88
	Software	32.99	1195.84	15125.20
8	PAB	33.90	248.42	4544.16
	Software	32.76	2807.39	35420.29

Key Takeaway PAB preserves strict priority guarantees under sustained overload.

Multiple Instance PAB

Latency and Instance Scaling

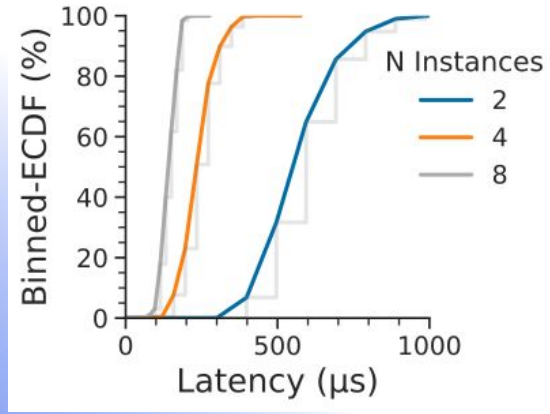


Key Takeaway PAB reduces latency as instances scale while maintaining zero loss, demonstrating effective horizontal scaling.

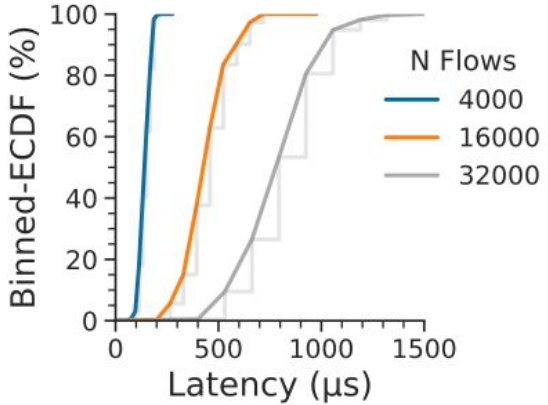
PAB Evaluation

Flow And Instance High Priority Latency

i) 4000 Flows



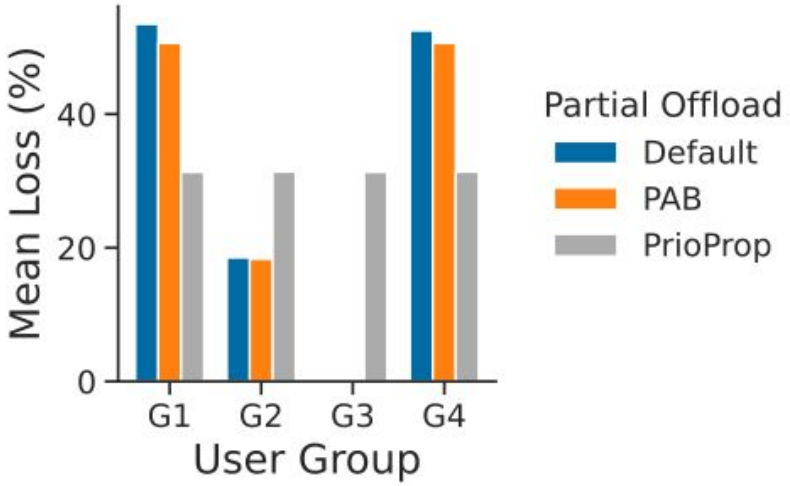
ii) 8 Instances



Key Takeaway Under a heavily overloaded scenario (32000 users/ 64000 software queues), HP latency remains constrained

PAB Evaluation

Loss Fairness



Key Takeaway PAB preserves the default scheduler's fairness behavior.

BISDN

Hybrid SW/NIC QoS needs coordination

Lack of backpressure breaks QoS guarantees

PAB is a lightweight, deployable fix in DPDK