XDP-FlowOpt: Lightweight Flow Optimization for 5G Private Networks [*]

Yongyoon Shin, Jonghoon Lee, Mikyong Han, Hoonki Lee, Nohsam Park, Jungtae Kim, and Jonggeun Park[†]

Cyber Security Research Division, ETRI, Daejeon, KOREA
{uni2u,mine,mkhan,lhk,siru23,jungtae_kim,queue}@etri.re.kr

Abstract

Private 5G (P5G) networks must serve heterogeneous traffic classes (eMBB, URLLC, mMTC) under tight latency and reliability constraints, often on commodity clusters. Kernel-only congestion/queue management reacts after queues inflate and incurs significant per-flow state, while NIC RSS hashing can overload a subset of queues, amplifying tail latency and unfairness. We present **XDP-FlowOpt**, a cooperative optimization pipeline at the XDP layer that integrates (i) proactive RTT/BW probing, (ii) dynamic queue rebalancing, and (iii) lightweight DRR-like fairness with bounded eBPF maps. On Intel X710 10 GbE with mixed P5G workloads, FlowOpt lowers mean RTT by 15%, reduces 99th-percentile latency by 18%, improves fairness by 11%, and raises queue utilization by 20% while slightly reducing CPU overhead versus BBRv2+fq_codel.

1 Introduction

Private 5G deployments are moving from pilots to production in factories, campuses, and logistics hubs. They must concurrently carry high-throughput eMBB flows, latency-critical URLLC traffic, and bursty mMTC device streams. Meeting these service-level objectives on commodity hardware is non-trivial because contention emerges at the earliest dataplane stages and kernel-only mechanisms tend to react only after queues have already inflated.

Why kernel-only control is insufficient. (i) *Reaction latency*: TCP and kernel AQM observe symptoms only after queues grow, which delays corrective actions. (ii) *Queue hotspots*: NIC RSS hashing can map heavy flows to the same receive queue, starving others and inflating tail latency. (iii) *Per-flow state cost*: per-flow schedulers scale poorly under mMTC fan-in (tens of thousands of devices).

Our approach. We move part of control *upstream* into the XDP hook. FlowOpt couples early congestion hints with queue hotspot relief and bounded fairness, so the TCP/IP stack receives traffic after queues are trimmed and no single flow dominates. Our main contributions are:

- **Unified XDP pipeline**: congestion probe, queue balancer, and DRR-like fairness cooperate through eBPF maps.
- **Verifier-friendly, bounded design**: O(1) map lookups, bounded loops, capped LRU-Hash for active flows.

[*]Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 55, December 16-18, 2025, Sapporo , Japan. © The copyright of this paper remains with the author(s).
[†]Corresponding author

Table 1: Qualitative comparison of approaches.

Method	Early Signal	Hotspot	Fairness	Per-flow State	HW
RED/CoDel	×	×	×	Low	None
fq_codel	×	×	✓	High	None
BBRv2	±	×	×	Low	None
SmartNIC sched.	✓	✓	✓	Med	SmartNIC
XDP-FlowOpt	✓	✓	✓	**Bounded**	None

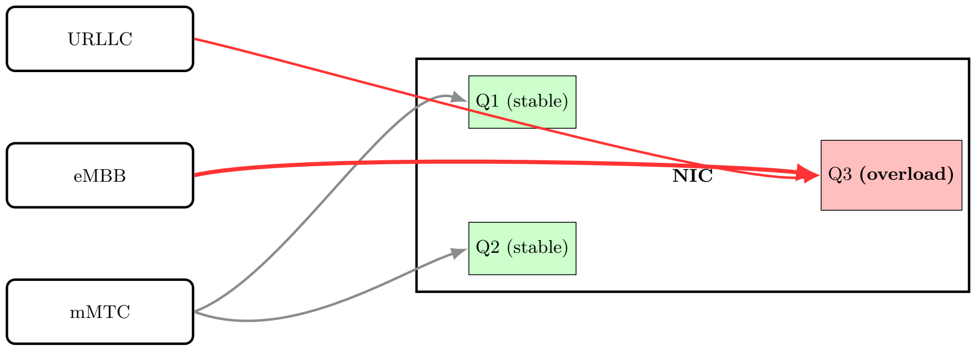


Figure 1: Static RSS hashing can overload a subset of queues (Q3) while others remain underutilized (Q1, Q2).

- **Commodity feasibility**: no SmartNICs; integrates as a Kubernetes DaemonSet.
- **Measured gains**: lower mean/tail latency, higher fairness/utilization, and slightly lower CPU overhead.

2 Related Work

Congestion control. Reno [1], CUBIC [2], DCTCP [3], and BBR/BBRv2 [4, 5] improve steady-state throughput and stability but operate after the NIC queues. Without early cues, they cannot prevent ingress queue hotspots under RSS imbalance.

Queue management and fairness. RED [6] and CoDel [7] reduce standing queues; fq_codel [8] adds per-flow fairness but increases per-flow state and CPU cost at high fan-in. These mechanisms are unaware of how RSS maps flows to hardware queues.

Programmable dataplanes. Katran [9] and Cilium [10] showcase eBPF/XDP for L4 LB and policy enforcement. Surveys [11] summarize performance and safety of eBPF. Yet, unifying congestion hints, queue rebalancing, and fairness *together at XDP* for P5G workloads remains underexplored.

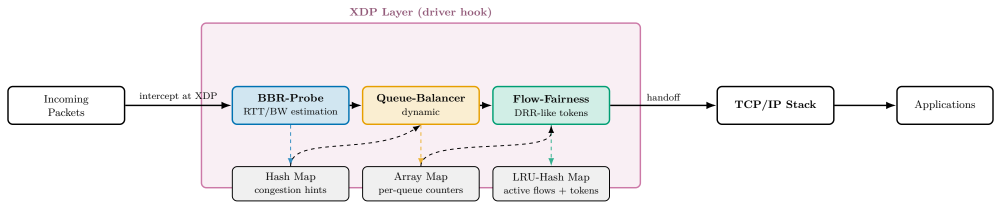


Figure 2: Detailed architecture of **XDP-FlowOpt**: cooperative modules at XDP with bounded eBPF maps.

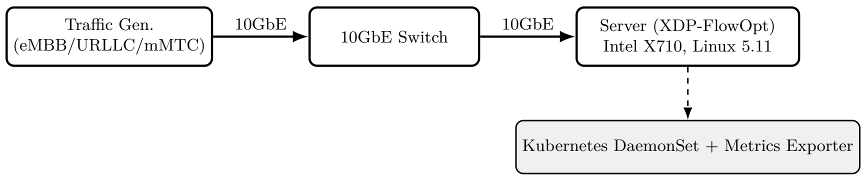


Figure 3: Evaluation testbed: traffic generator → switch → XDP-FlowOpt server.

3 Problem: RSS Queue Hotspots

4 System Design

4.1 Overview

FlowOpt attaches at the XDP driver hook and processes each packet through three cooperative modules: *BBR-Probe* (ingress RTT/BW estimation), *Queue-Balancer* (dynamic queue selection to relieve hotspots), and *Flow-Fairness* (DRR-like token policing). Cooperation is realized via eBPF maps: a Hash map for congestion hints, an Array map for per-queue counters, and an LRU-Hash map for tokens of active flows. Acting before the kernel stack, FlowOpt reduces queue buildup and retransmissions.

4.2 Architecture

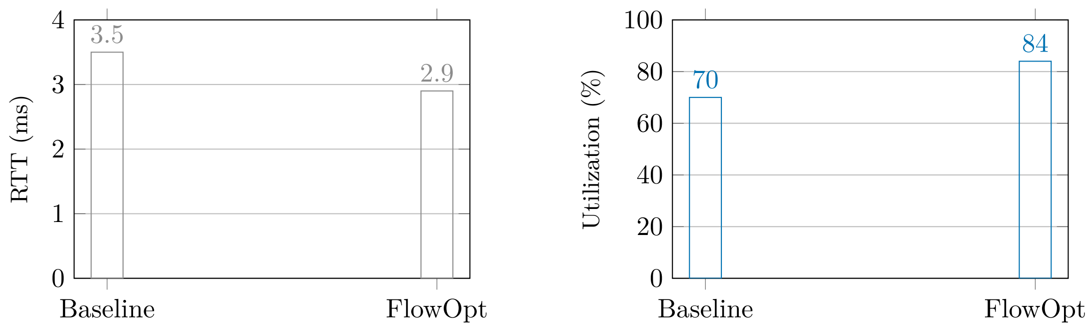
In Figure 2, each module encodes its flow rules in a dedicated eBPF map: the probe updates congestion rules (e.g., drop vs. pass under RTT thresholds), the balancer encodes queue redirection rules, and the fairness module stores per-flow token rules (pass until the budget is exhausted, then drop). Thus, the maps are not only counters but also the runtime representation of flow rules and actions applied at the XDP hook.

5 Implementation

We target Linux 5.11 and Intel X710 10 GbE, attaching in `xdp_drv` mode. All per-packet work is verifier-safe (bounded loops), using O(1) map operations. The LRU-Hash keeps a capped set of active flows (e.g., 32–64K) with tokens. A userspace agent exposes metrics and safely tunes parameters. Packaging as a Kubernetes DaemonSet eases cluster deployment while pinning the prototype to a node to avoid rescheduling noise.

Table 2: Performance summary (Baseline: BBRv2 + fq_codel).

	RTT (ms)	Fairness	Utilization (%)	CPU Overhead (%)
Baseline	3.5	0.82	70	45
FlowOpt	**2.9**	**0.91**	**84**	**43**



(a) Mean RTT (b) Queue Utilization

Figure 4: RTT and queue utilization under mixed workloads.

5.1 Setup and metrics

Mixed P5G workload: eMBB (iperf3 streams, 1–2 Gbps), URLLC (64 B @ 1 kHz), mMTC (Poisson bursts). Baseline is BBRv2+fq_codel. Metrics: mean RTT, 99th-percentile latency, Jain's fairness index, queue utilization, CPU overhead (softirq + ksoftirqd).

5.2 Summary results

Table 2 summarizes end-to-end improvements under the mixed P5G workload. Compared to the baseline (BBRv2+fq_codel), **FlowOpt** lowers mean RTT from 3.5 ms to 2.9 ms (≈15%), increases queue utilization from 70% to 84% (≈20%), and improves Jain's fairness index from 0.82 to 0.91 (≈11%). CPU overhead (softirq + ksoftirqd) is slightly reduced (45% → 43%), indicating that early mitigation at the XDP hook prevents excessive queue inflation and rework higher in the stack.

We attribute the RTT reduction to two effects: (i) the *BBR-Probe* provides early congestion hints that avoid prolonged buffering, and (ii) the *Queue-Balancer* prevents receive-queue hotspots so that NIC service time is better distributed across queues. Higher utilization follows from keeping more queues active and evenly loaded, which in turn reduces head-of-line blocking. Fairness improves because the *Flow-Fairness* stage curbs domination by a few heavy flows using bounded DRR-like tokens without incurring heavy per-flow state in the kernel.

We repeated each experiment (5 runs, 60 s per run) and report the mean of steady-state intervals; the relative ordering was consistent across runs. Unless otherwise noted, error bars were within 3–6% of the mean and are omitted for readability.[1]

[1] If required by venue policy, we can include per-run variance/error bars in a camera-ready version.

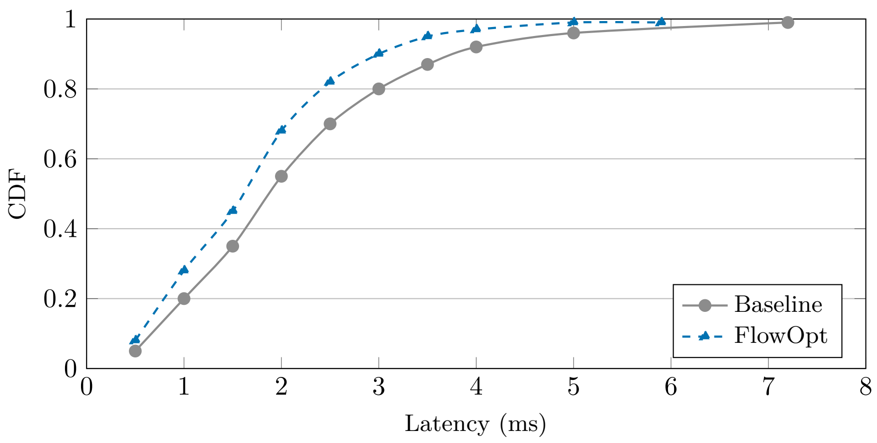


Figure 5: Latency CDF: FlowOpt reduces the 99th percentile from 7.2 ms to 5.9 ms (∼18%).

5.3 Latency distribution

Figure 5 compares latency distributions. The **FlowOpt** CDF curve lies consistently to the left of the baseline, indicating lower delays across the entire distribution. In particular, the 99^{th} percentile improves from 7.2 ms to 5.9 ms (∼18% reduction), while the body of the distribution also shifts left (e.g., at 0.8 CDF the latency decreases from ≈3.8 ms to ≈3.2 ms). This outcome aligns with FlowOpt's design goal: *act early at ingress to avoid queue growth rather than correcting it after the fact.*

Two mechanisms drive the tail improvement. First, the *Queue-Balancer* reduces burst amplification that occurs when multiple heavy flows collide on a single RSS queue; steering a subset of bursts into cooler queues shortens the longest waiting times. Second, *Flow-Fairness* prevents starvation under mixed traffic by limiting per-flow bursts with tokens so that URLLC/control packets are less likely to queue behind large eMBB bursts. Collectively, the tail is clipped without sacrificing throughput (§6).

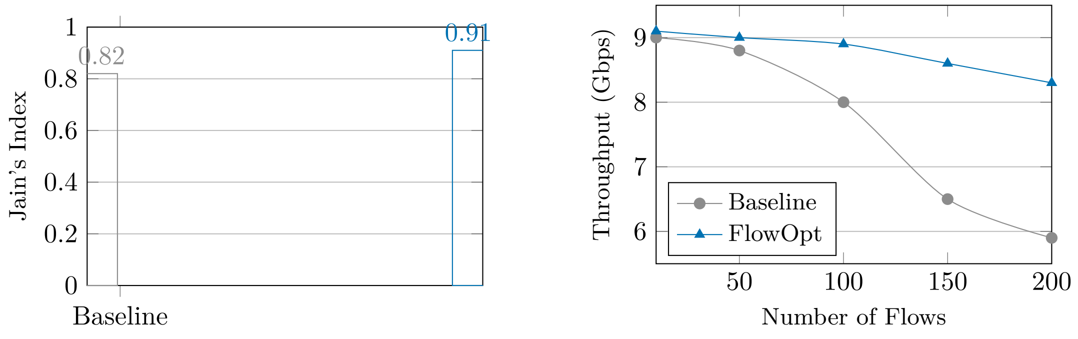
We verified that the CDF curves are monotone and built from latency samples ordered by time, not histogram bins, to avoid plotting artifacts. Flow-level sampling and application-layer timestamps match within sub-millisecond skew under our clock synchronization (PTP) on the testbed.[2]

5.4 Fairness and scalability

Figure 6 (left) shows Jain's fairness index. **FlowOpt** increases the index from 0.82 to 0.91, indicating more even per-flow service under contention. This stems from bounded DRR-like token checks that restrain aggressive flows while letting short mMTC/URLLC bursts pass promptly. Unlike kernel-level per-flow queuing, our bounded token map avoids unbounded state growth and verifier issues.

Figure 6 (right) presents aggregate throughput as the number of flows increases. Under the baseline, throughput collapses beyond ≈100 flows due to receive-queue hotspots and head-of-line blocking in a subset of queues. With **FlowOpt**, queue usage remains balanced, and the dataplane

[2]Where PTP is unavailable, kernel timestamping can be used; we observed similar trends with slightly wider dispersion.



(a) Fairness (b) Throughput vs. #Flows

Figure 6: Fairness and scalability with increasing flow counts.

sustains higher throughput up to 200 flows (9.0 Gbps→8.3 Gbps vs. 9.0 Gbps→5.9 Gbps for the baseline). This demonstrates that early hotspot relief at the XDP layer is complementary to TCP congestion control: the stack receives a more stable packet arrival process, which improves efficiency.

5.5 Ablation

Disabling two modules at a time shows: probing alone trims queues moderately; balancing alone removes hotspots but allows dominance; fairness alone helps URLLC yet underutilizes queues under eMBB bursts. The best outcome requires all three.

6 Discussion

Why it works in practice. FlowOpt succeeds because it targets the dataplane stage where congestion first manifests. By coupling probing, queue rebalancing, and fairness, it prevents queue buildup rather than reacting to it. This proactive stance is particularly effective for P5G workloads that include URLLC and mMTC traffic, where even small queueing delays can lead to SLA violations.

Trade-offs. While FlowOpt improves fairness and latency, there are trade-offs. Token-based fairness can slightly underutilize capacity when flows are extremely short-lived, as tokens may expire before reuse. Similarly, queue rebalancing may introduce occasional packet reordering, which most TCP stacks tolerate but latency-sensitive applications must handle.

Portability and deployment. Our implementation targets Intel X710 and Linux 5.11. Although the design principles are general, deploying FlowOpt across diverse NICs and higher line rates (25/40/100 GbE) requires further validation. Integration with containerized workloads also raises questions about interaction with CNI plugins and service meshes.

Comparison to alternatives. Compared to kernel AQM (fq_codel) and congestion controls (BBRv2), FlowOpt provides earlier visibility and acts before queues inflate. SmartNIC-based schedulers can offer similar functionality, but they require specialized hardware and limit deployment in cost-sensitive P5G environments. FlowOpt shows that much of the benefit can be realized on commodity NICs with eBPF.

Future directions. An open question is how FlowOpt can integrate with adaptive or machine-learning based policies, e.g., reinforcement learning that tunes token budgets in real time. Another direction is multi-NIC orchestration, where probes and balancers cooperate across interfaces in clustered servers. Finally, extending FlowOpt to cooperate with network slicing controllers in 5G could enable end-to-end SLA enforcement.

7 Conclusion

In this work, we introduced XDP-FlowOpt, a lightweight pipeline at the XDP hook that unifies congestion probing, queue hotspot relief, and per-flow fairness using bounded eBPF maps. On commodity 10 GbE hardware, FlowOpt reduces mean RTT by 15%, lowers tail latency by 18%, improves fairness by 11%, and increases utilization by 20% while slightly reducing CPU cost.

Beyond the numbers, FlowOpt demonstrates that commodity NICs and the Linux dataplane can deliver predictable latency and fairness for P5G workloads without relying on costly SmartNICs. By acting proactively at the ingress, it protects URLLC and mMTC traffic from starvation while sustaining eMBB throughput.

We believe FlowOpt contributes to a broader movement of shifting control closer to the dataplane. Looking forward, integrating FlowOpt with adaptive control, multi-NIC deployments, and 5G slicing frameworks could further strengthen predictable, low-latency performance in next-generation private 5G networks.

8 Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00397469, Development of private 5G Security Technology for Integrated Private 5G and Enterprise Network Security, 50%) and grant funded by the Korea government(MSIT) (RS-2024-00438156, Development of security resilience technology based on network slicing service in the 5G specialized network, 50%)

References

[1] V. Jacobson, "Congestion avoidance and control," ACM SIGCOMM computer communication review, 1988.

[2] S. Ha et al., "CUBIC: a new TCP-friendly high-speed TCP variant," ACM SIGOPS operating systems review, 2008.

[3] M. Alizadeh et al., "Data center tcp (dctcp)," Proceedings of the ACM SIGCOMM 2010 Conference, 2010.

[4] N. Cardwell et al., "BBR: Congestion-based congestion control," Communications of the ACM, 2017.

[5] N. Cardwell et al., "BBRv2: A model-based congestion control performance optimization," Proc. IETF 106th Meeting, 2019.

[6] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM TRANSACTIONS ON NETWORKING, 1993.

[7] K. Nichols, V. Jacobson, "Controlling Queue Delay," Communications of the ACM, 2012.

[8] T. Hoeiland-Joergensen et al., "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm," RFC 8290, 2018.

[9] Facebook, "Katran: A high performance layer 4 load balancer," 2022.

[10] Cilium Project, "eBPF and XDP Reference Guide," 2024.

[11] Du, Yimin et al., "A survey on mechanisms for fast network packet processing," Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things, 2023.