# Rethinking Telemetry Design for Fine-Grained Anomaly Detection in 5G User Planes

Niloy Saha, Noura Limam, Yang Xiao, and Raouf Boutaba
{n6saha, noura.limam, yang.xiao, rboutaba}@uwaterloo.ca, University of Waterloo, Canada

***Abstract*—Detecting QoS anomalies in 5G user planes requires fine-grained per-flow visibility, but existing telemetry approaches face a fundamental trade-off. Coarse per-class counters are lightweight but mask transient and flow-level anomalies, while per-packet telemetry postcards provide full visibility at a prohibitive cost that grows linearly with line rate. Selective postcard schemes reduce overhead but miss anomalies that fall below configured thresholds or occur during brief intervals.**

**We present *Kestrel*, a sketch-based telemetry system for 5G user planes that provides fine-grained visibility into key metric distributions such as latency tails and inter-arrival times at a fraction of the cost of per-packet postcards. *Kestrel* extends Count-Min Sketch with histogram-augmented buckets and per-queue partitioning, which compresses per-packet measurements into compact summaries while preserving anomaly-relevant signals. We develop formal detectability guarantees that account for sketch collisions, yielding principled sizing rules and binning strategies that maximize anomaly separability. Our evaluations on a 5G testbed with Intel Tofino switches show that *Kestrel* achieves $10\%$ better detection accuracy than existing selective postcard schemes while reducing export bandwidth by $10\times$.**

***Index Terms*—5G, QoS, Network Slicing, Network Telemetry**

## I. Introduction

Modern 5G and beyond networks leverage *network slicing* to provide dedicated logical networks for diverse applications, each with strict SLA requirements [1]–[3]. Data-plane components such as the User Plane Function (UPF) and O-RAN Centralized Unit (CU) aggregate traffic from multiple such slices onto hardware-accelerated platforms (e.g., P4 switches, SmartNICs, FPGAs). This makes them critical points for detecting QoS degradations that threaten slice-level SLAs.

**The monitoring dilemma.** Detecting QoS degradations in these data planes requires fine-grained, real-time telemetry. However, because thousands of flows share limited queues and on-chip memory, these systems typically only expose coarse per-class counters [4] or selective reports based on sampling or threshold triggers [5], which obscure transient or per-flow anomalies. Programmable hardware can, in principle, support full per-packet visibility through *postcard telemetry* that records queue depth, latency, and QoS identifiers for each packet. However, postcard export is prohibitively expensive: even at a modest 10 Gbps with 200 B packets, line-rate export would generate over 3 Gbps of telemetry (>$30\%$ overhead). This is impractical at production scale (100 Gbps+).

To make this challenge concrete, we focus on the UPF, a critical 5G data plane function that forwards user traffic over GTP-U tunnels identified by *tunnel endpoint identifiers (TEIDs)*. Each tunnel may carry multiple *QoS flows (QFIs)* mapped to standardized service classes and enforced by a small set of shared hardware queues. When multiple TEIDs share a queue, standard 3GPP per-QFI counters [4, 6] collapse fine-grained behavior into QFI-level averages. For example, counters polled every 1s may report a healthy 2ms average delay at the QFI level, yet per-TEID histograms can reveal a brief spike in TEID A, with $6\%$ of packets delayed >20ms. Postcards expose this, but at unsustainable export cost.

This illustrates the fundamental challenge: *how to obtain fine-grained (e.g., TEID-level) visibility under strict memory and bandwidth constraints?*

**Our approach.** Our key insight is that data plane anomalies at the UPF manifest as distributional signatures (§II-B): latency tails, sub-second spikes, per-TEID throughput violations, and oscillatory inter-arrival patterns. However, maintaining explicit per-TEID histograms is infeasible: each histogram requires multiple bins of state, which can quickly exhaust the limited memory available on hardware-accelerated data planes.

We present *Kestrel*, a telemetry system that provides per-TEID distributional visibility through a principled extension of Count-Min Sketch tailored to UPF anomaly detection. Unlike prior sketch-based systems [7, 8] that focus on volume metrics (packet counts, heavy hitters), *Kestrel* augments sketch buckets to capture compact histograms of latency and inter-arrival times – the distributional signals that distinguish different anomalies. To make distribution tracking robust, we develop a formal detectability framework that quantifies how sketch collisions and normal-traffic variation affect anomaly visibility. This analysis yields three practical design principles: (1) per-*queue identifier* (QID) sketch partitioning to limit cross-class interference, (2) adaptive binning that preserves sensitivity to rare events under normal load, and (3) principled sizing rules derived from anomaly detection requirements.

Our analysis shows that modest sketch dimensions ($w$=512, $d$=3) are sufficient to effectively detect diverse anomalies, including transient events as short as $50$ ms. *Kestrel* exports compact summaries ($\approx 0.15\%$ of user-plane traffic), achieving over $10\times$ *lower telemetry volume* than existing selective-postcard schemes [9], while improving anomaly detection accuracy by $10\%$ on mixed workloads.

**Contributions.** We make the following contributions:

- We characterize the telemetry requirements for UPF anomaly detection, showing that representative anomalies require per-TEID distributional visibility, including latency tails, inter-arrival patterns, and sub-second resolution (§II).
- We design a histogram-augmented sketch data structure that tracks per-TEID distributions using shared buckets rather

than per-flow state, enabling operation within the memory constraints of programmable switches (§III).
- We develop formal detectability guarantees for sketch-based anomaly detection, and derive principled sizing rules for sketch dimensions and bin placement that ensure weak anomalies remain visible despite measurement noise (§IV).
- We implement *Kestrel* on Intel Tofino switches, and demonstrate that it achieves $10\%$ better anomaly detection accuracy than selective postcard sampling while reducing export bandwidth by $10\times$ (§VI).

## II. Characterizing the Fine-Grained Telemetry Problem in 5G Data Planes

Modern 5G networks increasingly implement key data plane functions, including the User Plane Function (UPF) and O-RAN Centralized Unit (CU), on hardware-accelerated platforms [10]–[13]. These functions aggregate traffic from multiple network slices, which makes them critical points for detecting QoS degradations that threaten SLA compliance. We define any data plane event that threatens SLA compliance as an *anomaly*, whether it stems from malicious activity, misconfiguration, or disruptive traffic patterns.

The central challenge in detecting such anomalies is a fundamental *visibility gap*: these systems typically expose only coarse-grained counters or heavily sampled telemetry, which fail to capture the fine-grained, transient signals required for reliable detection. In UPFs, this gap arises from aggregation of per-session statistics; in CUs, it manifests as coarse event reports that obscure per-UE dynamics [14].

We focus on the UPF as a representative case, developing and evaluating our approach on a UPF testbed. However, the visibility gap and resulting telemetry requirements apply broadly across 5G data plane components. We first characterize the architectural and resource constraints that create this gap (§II-A), then demonstrate empirically the telemetry granularity required for reliable anomaly detection (§II-B).

### A. The Visibility Gap in UPF Telemetry

The visibility gap stems from a fundamental mismatch between QoS enforcement requirements and hardware constraints. UPFs forward traffic using GTP-U tunnels identified by TEIDs, with service classes distinguished by QFI and enforced by QERs. To sustain line-rate throughput, UPFs are increasingly offloaded to programmable hardware (e.g., P4 switches, SmartNICs) that provide typically 8-100 egress queues per port [10, 12, 13]. Thousands of active TEIDs must therefore multiplex onto these shared queues, creating QoS interdependencies: a single aggressive TEID can degrade performance for others sharing its queue, making per-TEID monitoring essential for anomaly attribution.

However, explicitly tracking per-TEID metrics is infeasible: programmable data planes provide only a few MB of SRAM, insufficient to maintain counters across thousands of concurrent TEIDs. Existing telemetry systems work around this as follows:

**Aggregate counters.** Standard 3GPP performance metrics [4], implemented in systems such as Open5GS [6], report per-QFI packet/byte counts, loss, and average delay every few seconds. By aggregating across TEIDs, they reduce state requirements but collapse per-packet dynamics into coarse aggregates that mask TEID-level variations.

**Postcard telemetry.** Programmable data planes can generate stateless per-packet metadata [5, 15, 16] containing queue depth, timestamps, and identifiers. Postcards avoid maintaining per-TEID state, but impose prohibitive export bandwidth: at 10 Gbps with 200 B packets, line-rate export exceeds 3 Gbps ($>30\%$ overhead). Recent systems adopt change-triggered sampling ($\Delta$-SMP) [9, 17], exporting telemetry only when changes in metrics exceed a threshold $\Delta$, but events that remain below thresholds or overlap with others are missed.

Neither approach provides what anomaly detection requires: *temporal granularity* to capture transient events, *spatial granularity* to isolate per-TEID behavior, and *distributional visibility* into queue dynamics beyond simple averages. Other telemetry systems face similar limitations (§VII). We demonstrate these requirements empirically next.

### B. Empirical Characterization of Anomaly Signatures

To validate these requirements and understand which telemetry signals enable reliable anomaly detection, we inject controlled anomalies into our Tofino-based UPF testbed, which carries a mix of application traffic: cloud gaming, live streaming, video conferencing, and IoT. Each application maps to standardized QFIs, with multiple TEIDs per application sharing queues. We implement eight queues[1] with strict-priority, weighted round-robin scheduling, and TrTCM-based policing [12, 13].

Figure 1 shows four representative anomaly types and their detection requirements. *Congestion* from sustained surges produces long-tail latencies concentrated on specific QFIs (Fig. 1a). *Policy abuse*, where TEIDs exceed QER allocations (e.g., by remapping to higher-priority classes), appears benign at the QFI level but reveals throughput violations and latency spikes when zoomed in to the TEID level (Fig. 1b). *Microbursts* from bursty video streams manifest as millisecond-scale spikes that vanish in one-second QFI aggregates but emerge at finer temporal resolution (Fig. 1c). *Contention* from competing backhaul traffic induces oscillatory inter-arrival patterns distributed across flows (Fig. 1d).

These anomalies share a critical property: *their signatures become visible only at appropriate measurement granularity*. Specifically, effective detection requires:

1) *Sub-second temporal resolution* to capture transient bursts.
2) *Per-TEID spatial resolution* to attribute misbehavior within shared queues.
3) *Distributional features* such as latency tails and inter-arrival patterns rather than simple averages.

The challenge is designing a telemetry primitive that preserves these signals while remaining lightweight enough for carrier-scale deployment.

## III. Design of *Kestrel*

### A. Design Objectives

Section II established that reliable anomaly detection requires per-TEID distributions at sub-second resolution without the memory overhead of explicit tracking or the bandwidth cost of

[1]Our Tofino-1 testbed operates at 10 Gbps with 8 egress queues per port.

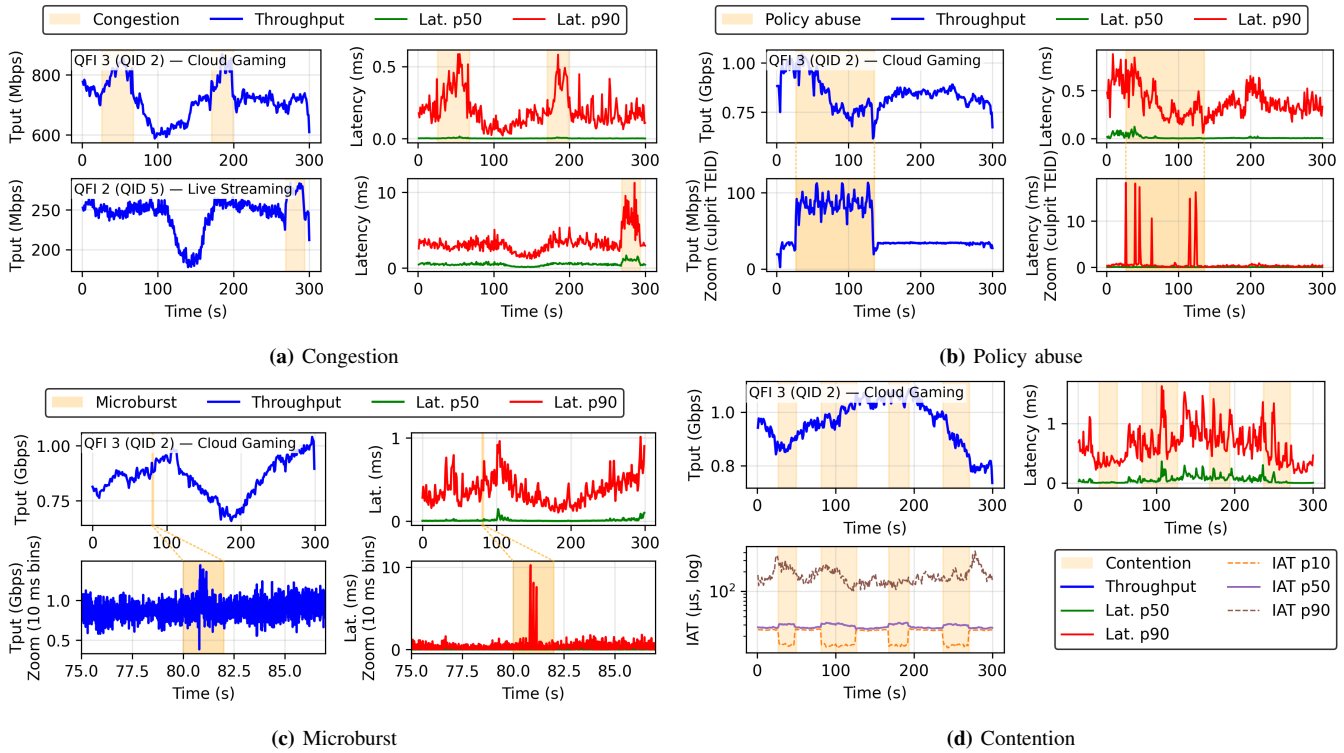



Fig. 1: Anomaly signatures require appropriate monitoring granularity. (a) Congestion produces sustained tail latency on specific QFIs. (b) Policy abuse appears benign in QFI aggregates but is visible at the culprit TEID. (c) Microbursts vanish in second-level averages yet emerge with sub-second resolution. (d) Contention induces distributed oscillations in inter-arrival times across flows.

continuous export. Addressing this challenge requires meeting three objectives simultaneously:

1) **Bounded export cost.** Telemetry overhead must remain fixed even as traffic rates and active TEIDs grow, eliminating the bandwidth explosion that makes postcard export untenable at carrier scale.
2) **Per-TEID distributional fidelity.** Telemetry must retain anomaly signatures such as latency tails, inter-arrival irregularities, and color counts, for each TEID, rather than collapsing behavior into slice-level averages (§II-B).
3) **Pipeline-constrained efficiency.** The telemetry primitive must fit comfortably within switch memory and execute within a single pipeline pass, avoiding recirculation, external memory, or coordination across stages. This ensures deployability on commodity programmable hardware, such as SmartNICs and Tofino switches, while leaving room for other 5G data plane functions.

### B. Design Challenges and Solutions

Meeting the objectives in §III-A is not straightforward. One natural alternative is to optimize selective postcard sampling, for example, with adaptive thresholds or smarter triggers. Such schemes fundamentally violate Objective (1): their export cost remains traffic dependent, flooding collectors precisely during bursts when visibility is most critical. We instead pursue sketch-based summaries that bound export volume regardless of traffic conditions. For this, we identify two main challenges:

**Challenge 1: From volumes to distributions.** State-of-the-art sketches such as ElasticSketch [7] and NitroSketch [8] excel at estimating flow volumes and identifying heavy hitters. They achieve high accuracy at scale, but remain tied to *volume metrics*: packet counts, byte counts, and flow sizes. What they cannot capture is how packets are distributed across latency or inter-arrival bins, the very signals that distinguish anomalies such as microbursts, contention, or policy abuse.

*Kestrel* addresses this gap by *extending Count–Min Sketch (CMS) [18]* to preserve distribution signals. CMS is an attractive foundation because it offers (i) clean analytical error bounds, (ii) a simple hash-and-increment update model that maps directly to ASIC pipelines, and (iii) a lightweight footprint that leaves memory and ALU budget for essential UPF functions such as policing, queuing, and scheduling. While CMS does not address overestimation bias, its simplicity makes it the right starting point.

To further mitigate bias, *Kestrel* maintains *per-QID sketches* rather than a single global structure (§III-C). Since QIDs group flows with similar QoS targets, partitioning reduces cross-class collisions and makes error bounds less pessimistic in practice. Each CMS bucket is *augmented* to hold not only packet and byte counters but also compact histograms of latency, inter-arrival times, and policing colors. These enriched buckets yield one-second summaries that preserve the distribution signals required for anomaly detection. The challenge lies not in augmenting the sketch buckets but in *configuring them correctly*: bin edges must cap diagnostic occupancy, and sketch width $w$ and depth $d$ must be sized to ensure anomalies remain visible despite collisions. We formalize these requirements in §IV, deriving detectability guarantees and sizing rules that make

histogram-augmented sketches practical.

**Challenge 2: Capturing timing without per-flow state.** Objective (2) also requires preserving inter-arrival time (IAT) patterns, which are crucial for distinguishing contention from congestion. A naïve design would timestamp each active flow, requiring $O(F)$ state for $F$ TEIDs, far beyond ASIC budgets and violating Objective (3). *Kestrel* instead timestamps *buckets*, not flows. Each CMS bucket stores the arrival time of its last packet. When the next packet (possibly from a colliding flow) arrives, the IAT is computed relative to this stored timestamp and recorded into the bucket's histogram. This reduces state to $O(d \cdot w)$ rather than $O(F)$. While collisions introduce some noise, our analysis (§IV) shows that the diagnostic bins remain sparse under baseline traffic, so anomalies remain visible.

Therefore, adapting sketches for UPF anomaly detection requires rethinking both the *signal model* (distribution vs. volume) and the *state model* (bucket vs. per-flow). We next present *Kestrel*'s data structure that realizes this design.

### C. Data Structure and Operations

**Architecture overview.** *Kestrel* combines two design choices that make sketching effective in the UPF setting.

First, *Kestrel* uses *per-QID partitioning*. Rather than one global sketch, *Kestrel* maintains a separate sketch per QID. Packets mapped to the same queue by UPF scheduling policies typically share QoS targets (e.g., low-latency cloud gaming vs. best-effort bulk). Per-QID partitioning reduces cross-class collisions and enables queue-specific binning (a 1ms latency threshold is meaningful for gaming but irrelevant for best-effort). While this increases the number of sketches, each sketch can be kept small: as we show in §IV and §V, sketches with only $w$=512 and $d$=3 suffice for robust anomaly detection. This makes per-QID partitioning practical, since the total memory cost remains well within hardware budgets.

Second, *CMS with enriched buckets* is employed to capture distributional signals. *Kestrel* maintains $d$ rows and $w$ buckets per row, with independent hash functions $h_1, \ldots, h_d$ over QoS flow keys $k = (\text{TEID}, \text{QFI})$. As shown in Figure 2, each bucket $j_i = h_i(k)$ stores four components: (a) *core counters* for packets and bytes, (b) *meter colors* tallying TrTCM outcomes (green, yellow, and red), (c) a *latency histogram* updated from per-packet sojourn time, and (d) an *IAT histogram* updated from the bucket's timestamp register. These enriched buckets allow *Kestrel* to summarize traffic distributions compactly.

**Packet updates.** On each arriving packet with key $k = (\text{TEID}, \text{QFI})$ and latency $t_s$, *Kestrel* computes the bucket indices $j_i = h_i(k)$ for each row $i$. For each corresponding bucket $(i, j_i)$, it then: 1) increments the packet and byte counters, 2) maps the latency $t_s$ to a bin using QID-specific edges, 3) computes the Inter-Arrival Time (IAT) as the difference from the bucket's last-seen timestamp, updates the corresponding IAT bin, and overwrites the timestamp, and 4) increments the color counter corresponding to the packet's TrTCM marking.

**Export and query model.** At the end of each 1s window, the control plane collects one compact record per bucket. To estimate the feature vector for a specific flow with key $k$, it queries all $d$ buckets $h_i(k)$ to which the flow was mapped. The estimate for each feature (e.g., the count in latency bin #3) is the minimum across the $d$ candidate buckets, following CMS query semantics. The control plane, which already maintains TEID–QFI mappings for bearer/session management, uses these mappings to reconstruct per-flow distributions suitable for ML-based anomaly detection. This model keeps the switch pipeline lightweight while ensuring anomaly-relevant signals are exposed at second-level cadence.

**Illustrative Example (Figure 2).** Consider a GTP-U packet arriving at the UPF with TEID=87 and QFI=2, experiencing a sojourn time of $25\mu$s and an inter-arrival gap of $18\mu$s. The UPF maps this flow to QID=3 according to its scheduling policies. *Kestrel* processes this packet as follows: First, it forms the sketch key $k$ from TEID+QFI and applies $d$ hash functions to map $k$ to buckets $(i, j_i)$ across all rows. Second, it retrieves the QID=3-specific bin edges; for latency ($\{0.5, 6.3, 82, \ldots, 4970\}\mu$s) and IAT ($\{11.5, 16.2, 22.9, \ldots, 2.7{\times}10^6\}\mu$s). Finally, for each bucket $(i, j_i)$, it: increments packet/byte counters; assigns the $25\mu$s latency to bin 2 (as shown in Fig. 2); computes and records the $18\mu$s IAT (updating bin 2 and refreshing the timestamp); and updates the color counter based on meters.

## IV. Parameterizing Kestrel

*Kestrel*'s design raises three practical questions: (i) how to ensure detectability when anomalies occupy only a small region of the distribution, (ii) how to size the sketch parameters to meet detection targets, and (iii) how to place bin boundaries and handle distribution drift. We address each in turn.

### A. Detectability Guarantees

We first establish when anomalies become visible. As discussed in §II, anomalies typically concentrate in certain regions of the latency or inter-arrival distributions. We designate these regions (the latency tail and IAT head bins) as the *diagnostic region* $T$. Let $N_T$ denote the number of packets in $T$ under normal traffic. Because $T$ is kept sparse (§IV-C), any anomaly-induced increase $\Delta_T$ in that region becomes readily detectable.

However, two factors can obscure this signal. First, only a fraction of anomaly packets may actually fall in $T$; we denote the spillover fraction by $\beta$ ($\beta$=0 means all land in $T$). Second, sketch collisions introduce noise on the order of $\varepsilon N_T$, where $\varepsilon{\approx}e/w$ is the CMS error rate. The following theorem establishes detectability while accounting for both effects:

**Detectability Condition.** For a flow $k$, an anomaly that adds $\Delta_T$ packets to its diagnostic bins $T$ and $\beta\Delta_T$ outside $T$ is detectable with probability at least $1-\delta$ when

$$\Delta_T \;>\; \frac{\varepsilon N_T x_k + (x_{k,T} + \varepsilon N_T)\,\varepsilon N'}{x_{k,\bar{T}} - \varepsilon N_T - \beta(x_{k,T} + \varepsilon N_T)},$$

where $x_k$ is the total mass of $k$ in the window, $x_{k,T}$ and $x_{k,\bar{T}}$ are its baseline masses in and outside $T$, $N'$ is the total packet mass, and $\varepsilon{\approx}e/w$ is the CMS error rate. (Proof in Appendix A.)

*Interpretation.* Detection succeeds when the anomaly lift $\Delta_T$ within $T$ exceeds the collision floor $\varepsilon N_T$ after accounting for spillover $\beta$. For example, a pkt/s and diagnostic cap $\rho$=1% ($N_T$=$10^4$), $w$=512 gives $\varepsilon N_T{\approx}53$ pkts. At $\beta$=0.3, the required $\Delta_T{\gtrsim}76$ pkts. Even a small 50ms microburst at 50kpps

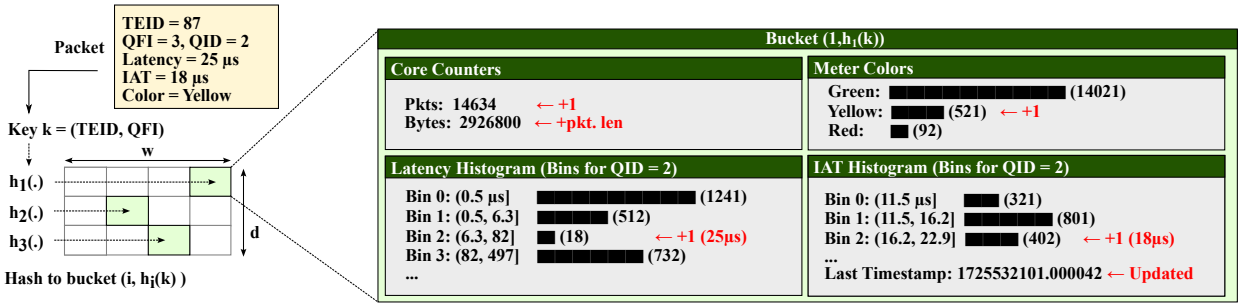



Fig. 2: **Data structure and operations of *Kestrel***. Buckets extend CMS to record packet/byte totals, latency histograms, IAT histograms, and policer colors. Example shows an arriving packet (TEID=87, QFI=3, QID=2, latency=25 µs, IAT=18 µs, and color=yellow) hashed to a bucket, and corresponding counters/bins updated (shown in red).

injects ≈2500 pkts, of which 70% ($\Delta_T \approx 1,750$) fall in $T$, well above the threshold.

### B. Choosing Width and Depth

The detectability condition translates directly into concrete sizing rules for the sketch parameters: width $w$ and depth $d$.

**Width Requirement.** To detect all anomalies that inject at least $\Delta_T^{\min}$ packets into the diagnostic region $T$, the sketch width must satisfy

$$w \geq \frac{e\, N_T^{\max}}{(1-\beta_{\max})\, \Delta_T^{\min}},$$

where $N_T^{\max}$ is the maximum diagnostic occupancy (set by binning) and $\beta_{\max}$ bounds the spillover fraction. For instance, with $N_T^{\max}=10^4$ and $\beta_{\max}=0.3$, detecting anomalies that inject $\sim$80 packets into $T$ requires $w \geq 485$; we use $w=512$ in practice, a configuration empirically validated in §VI-B.

**Depth Requirement.** Let $K=|T|$ be the number of diagnostic bins. If

$$d \geq \left\lceil \ln \frac{K+1}{\zeta} \right\rceil,$$

then, with probability at least $1-\zeta$, all CMS bounds needed for detection hold simultaneously within a window. In practice, $K$ is small (typically $2-3$ bins for latency tails and IAT heads). With $d=3$, these bounds hold with $> 99\%$ probability per window, and since most anomalies persist across multiple windows, the overall miss probability becomes negligible.

### C. Binning and Drift

Detectability depends not only on sketch width $w$ but also on the background occupancy $N_T^{\max}$ of the diagnostic region $T$. Because $N_T^{\max}$ is set by bin placement, careful binning is crucial: if too much baseline traffic falls in $T$, the collision noise $\varepsilon N_T$ grows, requiring larger widths to maintain the same detection guarantee.

**Target-Occupancy Binning.** *Kestrel* controls $N_T^{\max}$ using *target-occupancy binning*: bin edges are placed to cap baseline occupancy in $T$ at a target fraction $\rho$ (typically 1–2% of total packets). Setting the latency-tail boundary at the $(1-\rho)$-quantile and the IAT-head boundary at the $\rho$-quantile ensures $N_T^{\max}=\rho N$ (see Appendix for derivation). This design maximizes anomaly separability by keeping diagnostic bins nearly empty under normal conditions. We show that this strategy outperforms alternative binning schemes in §VI-D.

**Handling Drift.** Traffic distributions naturally drift over time, potentially increasing $N_T$ beyond $\rho N$. If left uncorrected, this raises the required width roughly in proportion to occupancy. To preserve sensitivity, the control plane monitors $N_T$ from exported sketches and triggers *re-binning* when $\hat{N}_T/N > \rho$, *during anomaly-free periods*. Our P4 implementation supports this dynamically: bin edges are stored as runtime-programmable table entries keyed by bin ranges and QID, allowing updates without pipeline disruption.

**Take-away.** Our analysis yields three configuration rules: (i) reserve a small diagnostic region ($2-3$ bins for latency tails and IAT heads); (ii) cap baseline occupancy using target-occupancy binning ($\rho \leq 2\%$) with drift adaptation; and (iii) set sketches to $w \approx 512$, $d=3-4$ under typical workloads. These parameters ensure $\Delta_T \gg \varepsilon N_T$ for practical anomalies, providing robust detection at modest memory cost. With these settings, *Kestrel* exports compact, anomaly-revealing summaries light enough for deployment in programmable UPFs.

## V. Implementation

**Testbed and Data plane.** We prototype *Kestrel*[2] on a UfiSpace S9180-32X switch equipped with an Intel Tofino ASIC. The switch implements UPF functions including queuing, scheduling, and TrTCM [19] metering, and connects over 10 GbE NICs to four Ubuntu 22.04 servers. These servers act as: (i) a GTP-U traffic generator that allows fine-grained control over TEID/QFI mappings, (ii) a control plane host that programs P4 tables and configures runtime telemetry, (iii) a collector for postcards and sketch registers, and (iv) an ML pipeline server for anomaly detection.

The data plane implementation consists of $\sim$2K lines of P4 in the egress pipeline. The design supports all three telemetry modes: 3GPP-PM counters, postcards (via packet mirroring), and sketches. It fits within a single pipeline pass ($\sim$7 stages) without recirculation. Sketches use $d=3$ rows and $w=512$ buckets (§IV); each bucket stores counters, 8-bin latency and IAT histograms, and TrTCM color tallies. Per-QID bin tables are runtime-programmable, enabling dynamic reconfiguration. Queues implement strict-priority and WRR scheduling across QIDs to emulate realistic UPF service classes.

**Workloads and Anomalies.** Traffic workloads combine a 5G operator trace [20] (cloud gaming, live streaming, buffered

[2]Source code available at https://github.com/nrg-uw/Kestrel.

TABLE I: Anomaly scenarios used in our evaluation: single anomalies (1–4) and mixed anomaly (5).

| Scenario | Anomaly | Target | Dur. | Freq. |
|---|---|---|---|---|
| 1 | Microburst | One/more TEIDs | $0.3-1.0$s | $10-30$s |
| 2 | Congestion | One/more QFIs | $25-40$s | $60-120$s |
| 3 | Contention | Many QFIs/TEIDs | $12-30$s | $90-150$s |
| 4 | Policy abuse | Selected TEIDs | $30-60$s | $150-300$s |
| 5 | Mixed | Multiple | As above | As above |

TABLE II: Telemetry approaches evaluated in our testbed, summarizing their switch operations, collection mechanisms, and asymptotic overhead.

| Baseline | Switch | Collector | Overhead |
|---|---|---|---|
| **3GPP-PM** | Per-QFI counters | Poll every 1s | *Low*, $O(\#\text{QFIs})$ |
| **Kestrel** | Per-QID sketches | Query every 1s | *Medium*, $O(wd)$ |
| **Δ-SMP** | Postcard sampling | Collect & aggregate every 1s | *High*, $O(\#\text{pkts})$ |

streaming) with additional traffic types we collected for IoT, VoIP, and best-effort applications. The generator sustains 1–4 Gbps aggregate load across up to 100 UEs × 9 QFIs (≈900 flows)[3]. We inject four representative anomaly scenarios: *microbursts* (short high-rate bursts), *congestion* (sustained overload), *contention* (shared backhaul bottlenecks), and *policy abuse* (QFI remapping), as well as mixed cases. Table I summarizes their scope, durations, and injection frequencies. Each anomaly ultimately degrades the QoS of affected UEs.

**Telemetry export and processing.** Sketches are exported every 1s via the Barefoot gRPC API, while postcards are captured by a dedicated collector with a memory-mapped ring buffer for high-throughput, zero-copy capture. All telemetry modes are aligned to one-second windows for fair comparison. Feature extraction proceeds according to the intrinsic visibility of each mode. Sketches yield per-(TEID,QFI) distributions by querying the $d$ rows for each key and applying CMS query semantics. Postcards provide per-(TEID,QFI) distributions directly from INT metadata (latency, IAT, color). Counters expose only QFI-level aggregates. From these, we construct feature vectors including traffic volume, latency percentiles and tail fractions, inter-arrival head fractions, policer color ratios, and contextual metrics such as TEID counts per QFI.

**Anomaly detection pipeline.** We implement an anomaly detection pipeline that incorporates four dedicated detectors targeting congestion, contention, microbursts, and policy abuse, respectively. Each detector is implemented as an XGBoost [21] model trained on baseline and anomaly-injected traces, enabling robust classification between normal and abnormal traffic patterns. As a result, each detector can focus on anomaly-specific behaviors. For instance, the microburst detector leverages sub-second temporal spikes, while the contention detector captures distributional shifts across flows. To ensure robustness, models are validated using temporally blocked cross-validation, preventing leakage between training and test windows.

## VI. Evaluation

We evaluate *Kestrel* to answer four questions:

1) How *accurately* can it detect diverse anomalies?
2) How *cost-effective* is it compared to existing approaches?
3) How *quickly* can it detect emerging anomalies?
4) Is it *practical* for deployment on programmable switches?

**Baselines.** We compare *Kestrel* against two representative telemetry approaches: (i) *3GPP-PM*, the standards-defined per-QFI performance measurements [4, 6], which expose per-QFI packet and byte counts, average delay, and loss; and (ii) Δ-*SMP*, selective postcard sampling that exports telemetry only when monitored metrics change beyond a configured Δ threshold [9, 17], thereby reducing postcard volume relative to per-packet export. Table II presents the key characteristics of these baselines alongside *Kestrel*, highlighting their switch operations, collection mechanisms, and asymptotic overhead. These baselines capture the design spectrum – from coarse-grained low-overhead (*3GPP-PM*) to fine-grained high-overhead (Δ-*SMP*).

We next evaluate these approaches along the four axes above: accuracy (§VI-A), cost-effectiveness (§VI-B), responsiveness (§VI-C), and practicality (§VI-D). For these experiments, we use the scenarios in Table I, sampling anomaly parameters uniformly from the listed ranges. The *frequency* column indicates inter-arrival times. We report metrics aggregated over 10 independent runs; where shown, error bars represent $95\%$ confidence intervals.

### A. Anomaly Detection Accuracy

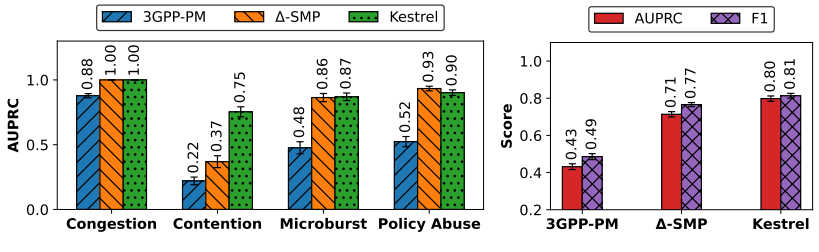
We first investigate how accurately *Kestrel* detects anomalies in both single- and mixed-anomaly scenarios (Table I).

**Single-anomaly scenarios.** Figure 3a compares the area under the precision–recall curve (AUPRC) across the three telemetry approaches for Scenarios 1–4. Detection accuracy varies significantly by anomaly type and telemetry granularity. For *congestion*, all three approaches perform well: *3GPP-PM* achieves $0.88$, while both Δ-*SMP* and *Kestrel* reach near-perfect accuracy (AUPRC ≈1.0). Congestion's clear QFI-level signature makes it detectable even with coarse counters.

*Contention* presents the starkest contrast: *3GPP-PM* collapses to $0.22$, Δ-*SMP* improves to $0.37$, while *Kestrel* achieves $0.75$, more than $3\times$ better than *3GPP-PM* and $2\times$ better than Δ-*SMP*. This gap stems from contention's anomaly signature: subtle distortions that evade both aggregate counters and threshold-based sampling but are captured by *Kestrel*'s sketches. For *microbursts*, the gap is narrower: *3GPP-PM* reaches $0.48$, while Δ-SMP ($0.86$) and *Kestrel* ($0.87$) perform comparably, both nearly doubling counter accuracy. For *policy abuse*, Δ-*SMP* slightly edges out *Kestrel* ($0.93$ vs. $0.90$), although both outperform counters ($0.52$).

**Mixed anomalies.** We next consider Scenario 5, where multiple anomaly types may overlap in the same window. This is a strictly harder setting: anomaly signatures dilute one another, and overlap increases the likelihood of false positives. Figure 3b shows that *3GPP-PM* fails entirely (AUPRC $0.43$, F1 $0.49$), while Δ-*SMP* achieves moderate accuracy ($0.71/0.77$). *Kestrel* maintains the highest performance ($0.80/0.81$), which is about $13\%$ higher AUPRC and $5\%$ higher F1 than Δ-*SMP*, and nearly $2\times$ better than *3GPP-PM*. *Kestrel*'s advantage stems

[3]Our prototype generator sustains a few Gbps; higher loads are feasible with a DPDK-based design, which we leave to future work.




**(a)** Single anomaly type (Scenarios 1–4). **(b)** Mixed anomalies (Scenario 5).

Fig. 3: **Detection accuracy across baselines**. (a) Single-anomaly scenarios (Scenarios 1–4): Continuous, distributional visibility (*Kestrel*) outperforms coarse counters and threshold-triggered sampling. (b) Mixed anomalies (Scenario 5): *Kestrel* sustains high AUPRC and F1, demonstrating robustness to concurrent anomaly types.

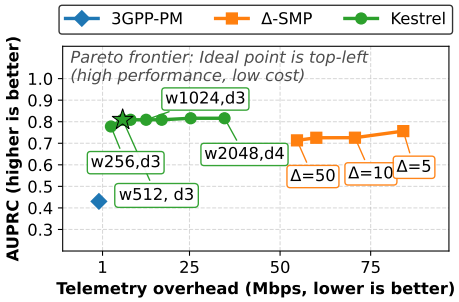



Fig. 4: **Pareto of AUPRC vs telemetry cost**. *Kestrel*'s best ($w$=512, $d$=3) reaches AUPRC 0.81 at ∼6 Mbps, 2× higher than *3GPP-PM* and 10× cheaper than Δ-*SMP*.

from continuous summarization of per-packet information, which allows it to capture even very subtle changes in telemetry patterns.

**Take-away.** *Kestrel* maintains high accuracy across anomaly types and remains robust to overlapping anomalies, outperforming baselines especially on subtle cases like contention.

### B. Cost-Effectiveness

We next turn from accuracy to cost, quantifying the telemetry overhead required to achieve the results above.

**Accuracy-cost trade-off.** Figure 4 plots the accuracy-cost trade-off, with ideal operating points in the top-left region. *3GPP-PM* occupies the lower-left: minimal overhead (∼1 Mbps) but poor accuracy (AUPRC ∼0.4). Δ-*SMP* achieves moderate accuracy (0.7–0.76) but at very high cost: 60–80 Mbps. *Kestrel* defines the Pareto frontier, achieving AUPRC 0.7–0.82 with only 5−10 Mbps export overhead. The sweet spot at $w$=512, $d$=3 delivers AUPRC 0.81 which is 2× higher than *3GPP-PM* and 10× cheaper than Δ-*SMP*. Larger configurations show diminishing returns, confirming our parameter analysis from §IV.

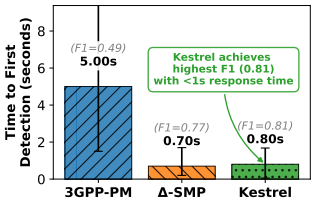
**Predictable and bounded cost.** At 1–4 Gbps tested loads (§V), *Kestrel*'s export rate remains stable at ∼6 Mbps (0.15–0.6% of traffic rate). Overhead depends only on configuration parameters ($w$, $d$, bin count, and number of QIDs) rather than the instantaneous packet rate, making telemetry export cost deterministic and controllable. In contrast, Δ-*SMP*'s event-driven postcards scale unpredictably with traffic burstiness and anomaly severity, producing spikes during busy periods. In our evaluation, Δ-*SMP* consumed 60–80 Mbps (1.5–8% of traffic rate), an order of magnitude higher despite lower accuracy.

**Take-away.** *Kestrel* provides high accuracy with predictable, configuration-bounded overhead, simplifying telemetry planning for operator-scale UPFs.

### C. Responsiveness

We next evaluate responsiveness, measured as the *time-to-first-detection* (TTFD): the median delay between anomaly onset and the first alarm. Figure 5 summarizes results across telemetry approaches (left) and per anomaly type (right), using the tuned thresholds from §VI-A.

**Aggregate performance.** Figure 5 (left) shows that *Kestrel* responds quickly (median 0.80s) with the highest detection




| Anomaly | 3GPP-PM | Δ-SMP | Kest. |
|---|---|---|---|
| Congestion | 6.00 | 1.20 | 1.05 |
| Contention | 6.00 | 0.20 | 1.05 |
| Microburst | 0.50 | 0.20 | 0.05 |
| Policy abuse | 4.00 | 1.20 | 0.55 |

Fig. 5: Median time to first detection (left: by scheme with F1 scores; right: by anomaly type).

performance (F1 0.81). Δ-*SMP* responds slightly faster (0.70s) but with lower performance (F1 0.77). *3GPP-PM* is much slower (5.00s) and far less accurate (F1 0.49), making it unsuitable for timely anomaly detection.

**Per-anomaly breakdown.** For *microbursts*, *Kestrel* responds immediately (0.05s median) while maintaining high F1 (0.92). Δ-*SMP* also reacts quickly (0.20s) but with lower accuracy, missing bursts when metric changes remain below the threshold. *3GPP-PM* responds reasonably fast (0.50s) but misses smaller events. Δ-*SMP* exhibits slightly lower latency in some cases (e.g., 0.20s for contention vs. 1.05s for *Kestrel*). This is because postcards are exported immediately upon threshold crossings, while *Kestrel* waits for the end of each window to query sketches. However, this speed advantage comes at the cost of detection fidelity: for *contention*, Δ-*SMP*'s AUPRC is only 0.37 (recall §VI-A) compared to *Kestrel*'s 0.75, demonstrating that faster alarms are only valuable when reliable. For *congestion*, both achieve similar latency (1.05–1.20s) with high accuracy. For *policy abuse*, sketches detect anomalies within 0.55s, roughly 7× faster than *3GPP-PM* (4.00s) and 2× faster than Δ-*SMP* (1.20s).

**Take-away.** *Kestrel* balances responsiveness and accuracy: while per-packet export can react instantly to anomaly events, *Kestrel*'s windowed summarization achieves comparable latency with higher detection fidelity.

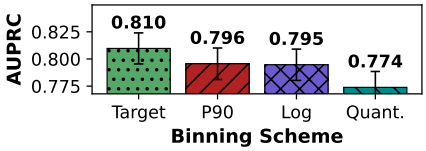
### D. Microbenchmarks and Design Insights

We next present microbenchmarks validating *Kestrel*'s practicality, covering (a) per-window processing latency, (b) binning sensitivity, and (c) Tofino resource usage (Fig. 6).

**Pipeline latency.** We measure end-to-end processing per 1s window, broken down into ingestion (I), feature extraction (F),

| Scheme | p50 (ms) | IQR (± ms) | I/F/ML (%) |
|---|---|---|---|
| 3GPP-PM | 1.83 | 0.04 | 37/29/33 |
| Kestrel | 53.3 | 0.45 | 27/72/1 |
| Δ-SMP | 200.4 | 7.3 | 33/66/1 |

**(a)** Per-window pipeline latency




**(b)** Accuracy across binning strategies

| Resource | Usage |
|---|---|
| Memory (SRAM/TCAM) | <20% |
| Hashing (Hash Units) | ≈25% |
| Computation (ALU) | ≈35% |

**(c)** Switch resource footprint

Fig. 6: Microbenchmarks of *Kestrel*. (a) Per-window pipeline latency, showing bounded cost. (b) Accuracy across binning strategies, target-occupancy performs best. (c) Switch resource footprint, showing modest usage.

and ML detection (ML). Figure 6(a) shows that all methods finish well within the deadline: counters in $\sim$2ms, *Kestrel* in 53ms, and $\Delta$-*SMP* in 200ms (median). Counters are trivially cheap ($\sim$2ms) but have poor performance. *Kestrel* spends most of its budget (72%) on feature extraction, as sketch registers must be queried and aggregated, yet the absolute cost remains modest: $\sim$ 53ms per window, independent of packet rate or number of flows. In contrast, $\Delta$-*SMP* spends two-thirds of its budget on feature extraction because each sampled packet is processed individually, causing latency to grow with traffic volume. *Kestrel* also exhibits low variance (IQR $\pm$0.45ms), confirming predictable performance. Therefore, it strikes a practical middle ground: richer than counters, yet an order of magnitude faster and more stable than postcards.

**Binning strategies.** A key design choice in *Kestrel* is how to discretize latency and IAT distributions into bins. We evaluate four approaches that represent different philosophies for capturing distributional behavior: (1) *Target*, *Kestrel*'s target-occupancy method (§IV-C), which caps baseline occupancy in diagnostic bins at $\rho$% to make anomalies appear as sharp spikes; (2) *P90*, which sets boundaries at the 90th and 99.8th percentiles with log spacing to emphasize the "knee" of latency/IAT distributions; (3) *Log*, which applies uniformly log-spaced bins across a fixed global range, capturing both short and long tails without per-QID tuning; and (4) *Quantile*, which divides samples into equal-frequency bins for uniform sensitivity across the range. Target-occupancy achieves the highest AUPRC (0.810) compared to P90 (0.796), Log (0.795), and Quantile (0.774). Although these improvements seem modest, they have a practical impact on anomaly detection: target-occupancy reduces contention false positives by $\sim$20% at 90% recall, compared to other binning approaches.

**Hardware usage.** Figure 6(c) shows *Kestrel*'s resource footprint on Tofino. The design consumes less than 20% of SRAM/TCAM, about 25% of hash units, and 35% of ALUs, leaving substantial headroom for other UPF functions. These results demonstrate that Kestrel fits comfortably within the constraints of a commercial programmable switch.

**Take-away.** *Kestrel*'s principled, distribution-aware design translates into practical gains: bounded latency, efficient resource use, and reliable anomaly detection on real hardware.

## VII. Related Work

**Telemetry paradigms.** *In-band telemetry (INT)* encodes metadata directly in packets [9, 22]–[25]. While widely used in datacenters, traditional INT is challenging to deploy in mobile networks due to GTP-U encapsulation, middlebox handling, and lack of end-to-end programmability. *Event-driven approaches* report telemetry only when certain predicates are met. For example, Marple [26], BurstRadar [27], and NetSeer [28] trigger telemetry on queue thresholds or on drops, reducing cost but limiting visibility to predefined events. *Sketch-based approaches* maintain compact summaries [7, 8, 29], and have been previously applied to scenarios such as heavy hitter and volumetric attack detection. However, prior works typically do not focus on *per-flow distributions* of latency, IAT, or policing colors, features that are critical for QoS anomaly detection at the UPF. *Kestrel* addresses precisely this gap.

**Anomaly detection.** Prior anomaly detection efforts focus on control-plane behavior and aggregate metrics. For the 5G core, approaches include deep sequence models to identify abnormal network function (NF) interactions [30], AI/ML for detecting anomalous traffic events [31], and detection of service degradation in cloud-based NF deployments [32]. In the RAN context, prior works have investigated distributed anomaly detection models across the edge and the cloud [33], as well as signaling storms and their mitigation [34]. Slice-level anomaly detection [35, 36] addresses security and QoS threats at the granularity of network slices. For example, [35] applies deep learning to detect DDoS attacks targeting slices and to dynamically isolate attackers, while [36] proposes a graph-based framework that correlates multivariate slice KPIs to detect and explain anomalies proactively. These approaches operate either on packet captures, incurring a very high cost, or on aggregate counters and KPIs that cannot capture transient per-TEID QoS violations within a slice. *Kestrel* complements these efforts by offering fine-grained data plane visibility, enabling the detection of QoS anomalies that precede or evade slice-level KPI deviations.

## VIII. Conclusion

We presented *Kestrel*, a principled telemetry system for detecting QoS anomalies in 5G user planes. By capturing distributional signals such as latency tails and inter-arrival patterns using compact, anomaly-aware sketches, *Kestrel* provides fine-grained visibility at minimal export cost. Our testbed evaluation shows that *Kestrel* delivers high detection accuracy, sub-second responsiveness, and predictable hardware utilization, demonstrating that anomaly-driven telemetry design is both practical and effective for next-generation mobile networks.

While this work focused on hardware-accelerated UPFs, we plan to extend the principles of anomaly-driven telemetry to software data planes (e.g., eBPF or VPP) in the future, enabling lightweight, anomaly-aware monitoring for heterogeneous 5G deployments.

## ACKNOWLEDGEMENTS

This work was supported in part by Rogers Communications Canada Inc., a Mitacs Accelerate Grant, and the Ontario Research Fund – Research Excellence program (Project# ORF-RE012-051) from the Province of Ontario. The views expressed herein are those of the authors and do not necessarily reflect those of the Province.

## REFERENCES

[1] NGMN Alliance. (2016) Description of network slicing concept. [Online]. Available: https://ngmn.org/wp-content/uploads/160113_NGMN_Network_Slicing_v1_0.pdf

[2] 3GPP, "System architecture for the 5g system; stage 2," 3GPP, Technical Specification (TS) 23.501, 09 2020, version 16.5.1.

[3] ——, "Service requirements for the 5g system," 3GPP, Technical Specification (TS) 22.261, 07 2024, version 18.14.0.

[4] ——, "Management and orchestration; 5G performance measurements," 3GPP, Technical Specification (TS) 28.552, 09 2020, version 17.0.0.

[5] ONF, "In-band network telemetry (int)," https://docs.sd-fabric.org/sdfabric-1.0/advanced/int.html, 2022, accessed: 2025-06-07.

[6] Open5GS. (2024) Open5GS github. [Online]. Available: https://github.com/open5gs/open5gs

[7] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: adaptive and fast network-wide measurements," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 561–575. [Online]. Available: https://doi.org/10.1145/3230543.3230544

[8] Z. Liu, R. Ben-Basat, G. Einziger, Y. Kassner, V. Braverman, R. Friedman, and V. Sekar, "Nitrosketch: robust and general sketch-based monitoring in software switches," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 334–350. [Online]. Available: https://doi.org/10.1145/3341302.3342076

[9] S. Sheng, Q. Huang, and P. P. C. Lee, "DeltaINT: Toward General In-band Network Telemetry with Extremely Low Bandwidth Overhead," in *Proceedings of the 29th IEEE International Conference on Network Protocols (ICNP)*, Nov. 2021, pp. 1–11.

[10] S. K. Singh, C. E. Rothenberg, J. Langlet, A. Kassler, P. Vörös, S. Laki, and G. Pongrácz, "Hybrid p4 programmable pipelines for 5g gnodeb and user plane functions," *IEEE Transactions on Mobile Computing*, vol. 22, no. 12, pp. 6921–6937, 2023.

[11] M. Rouili and R. Boutaba, "Blink: A p4-based 5g centralized unit," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2024, pp. 1–9.

[12] R. MacDavid, C. Cascone, P. Lin, B. Padmanabhan, A. ThakuR, L. Peterson, J. Rexford, and O. Sunay, "A p4-based 5g user plane function," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, ser. SOSR '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 162–168. [Online]. Available: https://doi.org/10.1145/3482898.3483358

[13] Z. Wen and G. Yan, "HiP4-UPF: Towards High-Performance comprehensive 5g user plane function on p4 programmable switches," in *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. Santa Clara, CA: USENIX Association, Jul. 2024, pp. 303–320. [Online]. Available: https://www.usenix.org/conference/atc24/presentation/wen

[14] M. Irazabal and N. Nikaein, "Tc-ran: A programmable traffic control service model for 5g/6g sd-ran," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 406–419, 2024.

[15] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "I know what your packet did last hop: Using packet histories to troubleshoot networks," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 71–85.

[16] Intel Corporation, "In-band network telemetry detects network performance issues," Intel, Tech. Rep., 2020, https://builders.intel.com/docs/networkbuilders/in-band-network-telemetry-detects-network-performance-issues.pdf.

[17] S. R. Chowdhury, R. Boutaba, and J. François, "Lint: Accuracy-adaptive and lightweight in-band network telemetry," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 349–357.

[18] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196677403001913

[19] J. Heinanen and R. Guerin, "A two rate three color marker," Internet Requests for Comments, RFC 2698, 1999. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2698.txt

[20] Y.-H. Choi, D. Kim, and M. Ko. (2023) 5g traffic datasets. [Online]. Available: https://dx.doi.org/10.21227/ewhk-n061

[21] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[22] The P4.org Application Working Group, "In-band Network Telemetry (INT) Dataplane Specification," 2024, https://p4.org/p4-spec/docs/INT_v2_1.pdf.

[23] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "Hpcc: high precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 44–58. [Online]. Available: https://doi.org/10.1145/3341302.3342085

[24] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "Pint: Probabilistic in-band network telemetry," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 662–680. [Online]. Available: https://doi.org/10.1145/3387514.3405894

[25] S. Tang, S. Zhao, X. Pan, and Z. Zhu, "How to Use In-Band Network Telemetry Wisely: Network-Wise Orchestration of Sel-INT," *IEEE/ACM Transactions on Networking*, pp. 1–15, 2022.

[26] S. Narayana, A. Sivaraman, V. Nathan, P. Goyal, V. Arun, M. Alizadeh, V. Jeyakumar, and C. Kim, "Language-directed hardware design for network performance monitoring," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 85–98. [Online]. Available: https://doi.org/10.1145/3098822.3098829

[27] R. Joshi, T. Qu, M. C. Chan, B. Leong, and B. T. Loo, "Burstradar: Practical real-time microburst monitoring for datacenter networks," in *Proceedings of the 9th Asia-Pacific Workshop on Systems*, ser. APSys '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3265723.3265731

[28] Y. Zhou, C. Sun, H. H. Liu, R. Miao, S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi, P. Zhang, D. Cai, M. Zhang, and M. Xu, "Flow event telemetry on programmable data plane," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 76–89. [Online]. Available: https://doi.org/10.1145/3387514.3406214

[29] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13. USA: USENIX Association, 2013, p. 29–42.

[30] Y. Tan, J. Liu, Y. Li, and J. Wang, "Deep learning based proactive anomaly detection for 5g core control plane network function interactions," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2025.

[31] A. Mekrache, K. Boutiba, and A. Ksentini, "Combining network data analytics function and machine learning for abnormal traffic detection in beyond 5g," in *Proceedings of the IEEE Global Communications Conference*, 2023, pp. 1204–1209.

[32] F. Michelinakis, J. S. Pujol-Roig, S. Malacarne, M. Xie, T. Dreibholz, S. Majumdar, W. Y. Poe, G. Patounas, C. Guerrero, A. Elmokashfi, and V. Theodorou, "Ai anomaly detection for cloudified mobile core architectures," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1976–1992, 2023.

[33] C. Sun, U. Pawar, M. Khoja, X. Foukas, M. Marina, and B. Radunovic, "Spotlight: Accurate, explainable and efficient anomaly detection for open ran," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. ACM, Nov. 2024.

[34] A. Tabiban, H. A. Alameddine, M. A. Salahuddin, and R. Boutaba, "Signaling storm in o-ran: Challenges and research opportunities," *IEEE Communications Magazine*, vol. 62, no. 6, pp. 58–64, 2024.

[35] B. Bousalem, V. F. Silva, R. Langar, and S. Cherrier, "Ddos attacks detection and mitigation in 5g and beyond networks: A deep learning-based approach," in *Proceedings of the IEEE Global Communications Conference*, 2022, pp. 1259–1264.

[36] A. Chawla, A.-M. Bosneag, and A. Dalgkitsis, "Graph-based interpretable anomaly detection framework for network slice management in beyond 5g networks," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–6.

# APPENDIX

## APPENDIX A PROOFS

This appendix provides proofs for the formal guarantees and sizing rules in §IV.

**Proof of Detectability Condition (with spillover).** By the CMS guarantee, each bin estimate satisfies $x_{k,j} \leq \hat{x}_{k,j} \leq x_{k,j} + \varepsilon N_j$ with probability at least $1 - \delta$. Summing over the diagnostic bins $T$ gives:

$$x_{k,T} \leq \hat{x}_{k,T} \leq x_{k,T} + \varepsilon N_T,$$

and summing over all bins gives:

$$x_k \leq \hat{x}_k \leq x_k + \varepsilon N'.$$

In baseline conditions, the maximum estimated diagnostic ratio is

$$\frac{\hat{x}_{k,T}}{\hat{x}_k} \leq \frac{x_{k,T} + \varepsilon N_T}{x_k}.$$

During an anomaly adding $\Delta_T$ packets to $T$ and $\beta\Delta_T$ elsewhere, the diagnostic mass increases while the total mass grows modestly. The minimum estimated ratio becomes

$$\frac{\hat{x}_{k,T}}{\hat{x}_k} \geq \frac{x_{k,T} + \Delta_T}{x_k + \Delta_T + \beta\Delta_T + \varepsilon N'}.$$

Detection succeeds when the anomaly-window ratio exceeds the baseline maximum. Rearranging this inequality yields the detectability condition. □

**Proof of Width Requirement.** In the sparse diagnostic regime ($x_{k,T} \ll x_k$, $N_T \ll N'$), the detectability condition simplifies to

$$(1 - \beta)\Delta_T \gtrsim \varepsilon N_T, \quad \text{where } \varepsilon \approx e/w.$$

Enforcing the design constraints $N_T \leq N_T^{\max}$, $\Delta_T \geq \Delta_T^{\min}$, and $\beta \leq \beta_{\max}$ gives

$$(1 - \beta_{\max})\Delta_T^{\min} \geq \frac{e}{w} N_T^{\max}.$$

Solving for $w$ yields the width requirement. □

**Proof of Depth Requirement.** Each CMS estimate violates its error bound with probability $\leq e^{-d}$. Detection involves $K$ diagnostic-bin estimates and one total-count estimate ($K+1$ events total). By union bound, all bounds hold simultaneously with probability $\geq 1 - (K+1)e^{-d}$. Setting this $\geq 1 - \zeta$ gives

$$d \geq \left\lceil \ln\left(\frac{K+1}{\zeta}\right) \right\rceil.$$

□

**Proof of Target-Occupancy Proposition.** Let $F$ be the baseline CDF. Placing the latency-tail boundary at the $(1-\rho)$-quantile ensures exactly $\rho N$ packets have latency above that boundary. Similarly, placing the IAT-head boundary at the $\rho$-quantile ensures $\rho N$ packets have IAT below it. Thus, $N_T^{\max} = \rho N$. □

**Proof of Drift Effect Proposition.** From the width requirement, required width scales as $w \propto N_T^{\max}$. If diagnostic occupancy increases from $\rho N$ to $\rho' N$, then $w_{\text{new}}/w_{\text{old}} = \rho'/\rho$. □

**Practical Implications.** These results explain the parameter choices in §IV-B: width $w$ scales linearly with $N_T^{\max}$ but inversely with $\Delta_T^{\min}$, while depth $d$ grows only logarithmically with the number of diagnostic bins $K$. This justifies why moderate parameters ($w$=512, $d$=3) suffice in practice, as empirically validated in §VI-B.