Fast-Path In-Network Inference on Intel Infrastructure Processing Units

Aristide Tanyi-Jong Akem
University of Southampton, United Kingdom
a.t-j.akem@soton.ac.uk

***Abstract*—SmartNICs enable low-latency in-network machine learning (ML) inference; however, existing approaches are constrained by limited or absent P4 programmability and inadequate support for stateful inference in the fast path. The emergence of Intel Infrastructure Processing Units (IPUs) offers a new opportunity to combine P4-programmable fast-path processing with on-board compute capabilities. In this demo, we showcase the first in-network ML inference system on Intel IPUs using P4. We map decision tree models to match-action pipelines in the fast path and augment them with stateful features computed on IPU cores and integrated at runtime. Our system achieves real-time, wire-speed classification with up to 99% accuracy while maintaining low latency comparable to L2 forwarding.**

***Index Terms*—In-network inference, IPU, machine learning, P4**

I. Introduction

Programmable data planes have emerged as a key enabler for high-performance in-network processing, allowing custom packet-level operations to be executed at line rate via domain-specific languages such as P4 [1]. This capability has spurred interest in in-network machine learning (ML), where inference tasks are offloaded directly to network devices to achieve low latency [2], [3]. Consequently, SmartNICs are increasingly deployed to offload ML and other complex infrastructure functions, such as security and encryption, load balancing, and traffic classification, thereby reclaiming host CPU cycles while maintaining execution flexibility [4].

Despite these advancements, existing SmartNIC-based ML inference systems face a fundamental trade-off. One class of systems utilizes general-purpose cores or dedicated on-board accelerators to perform inference; however, these *slow-path* approaches often struggle to sustain true line-rate processing due to hardware bottlenecks [5], [6]. Conversely, designs that leverage programmable packet-processing pipelines for *fast-path* inference are largely restricted to stateless models and per-packet features, constrained by the rigid memory and processing limits of the data plane [7]. This reveals a critical architectural gap: current approaches fail to coordinate fast-path pipelines with on-board compute resources. Specifically, they lack support for hybrid stateless-stateful inference that spans both the fast packet-processing pipeline and general-purpose cores, a capability now achievable through emerging platforms like Intel Infrastructure Processing Units (IPUs) [8].

In this demo, we present the first in-network ML inference implementation on Intel IPUs using P4. Our approach maps tree-based models to the IPU match-action pipeline by binarizing features prior to model training and encoding root-to-leaf paths as ternary match rules, which enables efficient wire-speed inference in the fast path. To overcome the constraints of stateless processing, we augment this design with a hybrid architecture where mirrored packets are processed by on-board IPU cores to compute stateful features. These features are then dynamically injected back into the data plane for use in subsequent classification. We demonstrate real-time classification using high-speed traffic generated via `PktGen-DPDK` and publicly available PCAP traces, achieving high accuracy while maintaining latency comparable to standard L2 forwarding.

Our work differs from prior in-network ML research in two key aspects. First, unlike switch-based systems, we leverage the unique architecture of IPUs to tightly couple programmable fast-path processing with general-purpose compute, which enables more flexible inference and support for more complex features. Second, in contrast to existing SmartNIC-based ML systems, which often lack P4 programmability or rely exclusively on slow-path execution, our design achieves line-rate inference while incorporating stateful, flow-aware features through coordination between the fast and slow paths.

This demo complements our paper [9], appearing in the IEEE NetSoft 2026 main conference, which focuses on hybrid operation, combining fast-path inference with feature computation in the cores. The demo extends it by highlighting the broader IPU in-network ML design space, showcasing both pure stateless inference, executed entirely in the fast path, and hybrid (stateless + stateful) inference, where stateful features are computed on the on-board cores and integrated into the fast path at runtime for more fine-grained inference.

II. System Architecture

The architecture of the hybrid inference system is shown in Figure 1. It is built on an Intel IPU that integrates a P4-programmable fast path with on-board processing cores. The design enables both stateless and hybrid stateful inference through a coordinated interaction between the data plane and the processing cores. To accommodate the strict data-plane constraints, namely limited on-chip memory, restricted arithmetic expressiveness, and bounded pipeline depth, we adopt binary decision trees (BDTs) for fast-path ML deployment in P4. Their comparison-based structure maps naturally onto match-action semantics, enabling efficient deployment within the pipeline without requiring complex state or computation.

Packets first enter the P4 pipeline, where stateless features f_{SL} are extracted from packet header fields $h_{1..n}$. These features are immediately available for fast-path inference using the binary decision tree (BDT) table. In the stateless-only mode, classification is performed entirely at this stage, enabling line-rate inference without involving the cores.

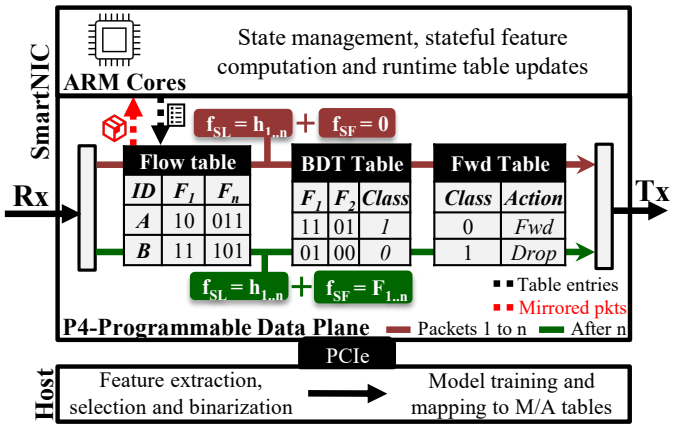

Figure 1: System overview. f_{SL} and f_{SF} respectively denote stateless and stateful feature sets, while $h_{1..n}$ and $F_{1..n}$ are respectively the stateless packet header fields, and the stateful features retrieved from the flow table.

In parallel, the first n packets of each flow are mirrored to the IPU cores, as indicated by the red path in Figure 1. The cores use these packets to compute and aggregate flow-level statistics, constructing the stateful feature set f_{SF}. This process implements early flow classification, where a flow is classified after observing its first n packets. Upon receiving the n-th packet, the cores finalize the computation of the stateful features and inject them into the data plane by installing a corresponding entry in the flow table. This update, shown by the black arrow in the figure, makes the computed features available to the fast path for subsequent packets of the flow.

From the $(n+1)$-th packet onward, the processing differs from the initial phase. Unlike the first n packets, which do not find matching entries in the flow table and are therefore mirrored to the cores, these subsequent packets match against the installed flow table entry and retrieve the precomputed stateful features $F_{1..n}$. These features are then combined with the stateless features and used as input to the BDT table, enabling hybrid inference that incorporates both per-packet and flow-level information. In both stateless-only and hybrid modes, the classification result produced by the BDT table is subsequently matched on a forwarding (Fwd) table. This table determines the final action applied to the packet, such as forwarding or dropping, based on the inferred class label.

As in prior work on ML inference in the data plane [2], model preparation is performed outside the fast path, in our case on the host, where both training and mapping are carried out. Raw packet traces are processed to extract stateless header features and to compute additional stateful features over the first n packets of each flow, after which the selected features are binarized and the BDT model is trained using Scikit-Learn. To map the trained model onto a match-action table, each root-to-leaf path is translated into a sequence of bits representing the binarized features and encoded as value-mask pairs, which are then converted into ternary match rules. This results in a one-to-one mapping between leaf nodes and table entries with associated class labels, enabling a compact representation.

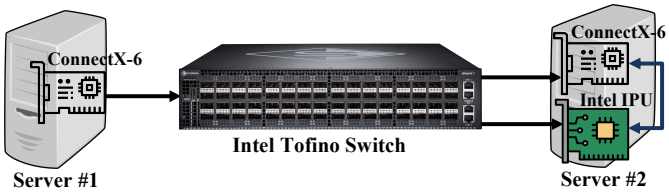

Figure 2: Testbed setup. The two servers are equipped with Mellanox ConnectX-6 NICs and are connected via an Intel Tofino switch. Fast-path ML inference runs on the Intel IPU in Server #2, where the NIC and IPU are also interconnected.

III. Demonstration

We demonstrate the operation of the fast-path ML inference framework for stateless and hybrid attack detection and classification, measuring key performance metrics.

A. Experimental setup

Figure 2 illustrates the demonstration setup. All experiments are conducted using an Intel Infrastructure Processing Unit (IPU) Adapter E2100-CCQDA2 [8], a PCIe 4.0 ×16 SmartNIC equipped with two 100 GbE ports and a P4-programmable packet-processing pipeline capable of sustaining line-rate traffic. The IPU integrates an on-NIC compute complex with 16 Arm cores and high-bandwidth on-board memory, enabling concurrent data-plane processing and control-plane execution, as described in §II. The IPU is installed in Server #2, which is equipped with an Intel Xeon Silver 4510 processor clocked at 2.4 GHz and 375 GB of DDR4 memory, and also hosts a dual-port Mellanox ConnectX-6 100 GbE NIC that is interconnected with the IPU. Another server (Server #1), equipped with an AMD EPYC 7443P 24-Core Processor clocked at 2.85 GHz, 251 GB of DDR4 memory, and a similar 100 GbE NIC, is used to inject traffic. The two servers are connected via an Intel Tofino switch, over which traffic is forwarded from Server #1 to Server #2. We use `tcpreplay` to replay packet traces at their original rates and `PktGen-DPDK` for high-speed tests.

B. Dataset and metrics

We evaluate the IPU-based packet classification system on the widely used ToN-IoT [10] dataset. It contains benign traffic and multiple cyber-attack scenarios collected from a representative medium-scale testbed spanning cloud, fog, and edge domains. It includes traffic generated by a diverse set of IoT and IIoT devices, such as sensors and actuators, as well as non-IoT endpoints, and covers a range of attack types observed in operational deployments, including ransomware, scanning, injection, and denial-of-service. The task is to classify traffic into one of seven classes, *i.e.*, six attack types and benign traffic. Separate models are trained and deployed for the stateless-only and hybrid approaches.

We evaluate the classification performance of the deployed models using the F1 score together with true positive and false positive rates, computed as follows: $\mathrm{F1} = \frac{2TP}{2TP+FP+FN}$, $\mathrm{TPR} = \frac{TP}{TP+FN}$, and $\mathrm{FPR} = \frac{FP}{FP+TN}$. For each of the metrics, we report both the macro and weighted averages.

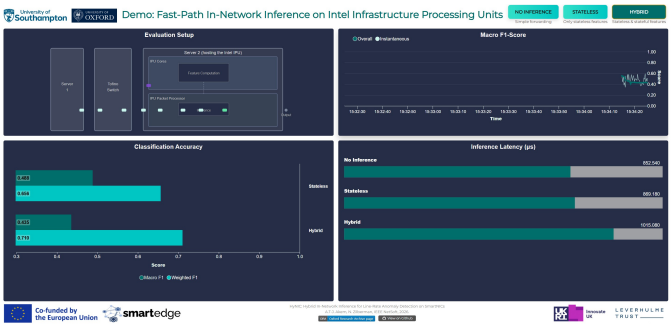

Figure 3: Visualization dashboard

C. Demonstration workflow and evaluation

1) Workflow: The demonstration is driven from a demo desk, where we remotely access the two servers hosting the packet generator/replayer and the Intel IPU. In the stateless mode, replayed packets are processed entirely within the IPU data plane, where header-extracted features are matched directly onto the BDT table and classification decisions are made per packet. To validate correctness, classification results are embedded in the time-to-live (TTL) field of the IP header during processing. The packets are then captured on Server #2 and analyzed to extract and verify the predicted labels.

In the hybrid setting, the workflow extends to incorporate stateful inference. Early packets are mirrored to the IPU cores, where stateful features are computed over the first n packets of each flow and maintained in per-flow state. These features are then injected into the flow table in the data plane, where they are combined with stateless header features for classification of subsequent flow packets. The following information is displayed on a dashboard as shown in Figure 3:

- Different packet classification workflows
- Visualization of classification performance
- Monitoring of system performance metrics
- Interactive switching between inference modes

2) Classification results: Table I reports the average performance across all classes for both the stateless per-packet model and the hybrid inference model. The results show that the hybrid approach significantly improves macro-level performance, with notable gains in TPR and F1-score, while reducing FPR. In particular, the hybrid approach achieves stronger detection performance across multiple attack classes, demonstrating the benefit of incorporating stateful features computed on the IPU cores.

3) Throughput: To evaluate system performance, we use `PktGen-DPDK` to generate high-speed traffic and replay benign ToN-IoT traces via the Tofino switch, measuring throughput on Server #2 using DPDK port statistics. The IPU consistently achieves near line-rate performance without packet loss across workloads. Throughput reaches 95–99 Gbps for larger packet sizes (1024 B, 512 B), effectively saturating the 100 Gbps link. Throughput remains high at smaller sizes, with up to 96 Gbps at 44 Mpps for 256 B packets. Similar near line-rate performance is observed during PCAP replay, confirming robust performance under realistic traffic conditions.

	Macro Average			Weighted Average		
Model	**TPR**	**FPR**	**F1**	**TPR**	**FPR**	**F1**
Stateless	0.3839	0.0643	0.3976	0.6029	0.0532	0.6507
Hybrid	0.6115	0.0213	0.6182	0.8763	0.0256	0.8799
Gain	+0.2276	-0.0430	+0.2206	+0.2734	-0.0276	+0.2292

Table I: Model performance for stateless and hybrid inference

4) Latency: We measure end-to-end latency as the time from packet transmission to inference completion using embedded transmit timestamps and kernel-level receive timestamps on Server #2, avoiding synchronization errors. This latency captures forwarding, feature extraction, and inference overhead. Results show a stable median latency of 130–145 μs for fast-path inference. Compared to an L2-Fwd baseline of 131.41 μs, the IPU-based inference pipeline adds negligible overhead, demonstrating that real-time inference can be achieved with low and predictable latency.

IV. Concluding Remarks

This demonstration highlights the Intel IPU as an effective platform for inline, in-network ML inference, combining a P4-programmable fast path with on-board processing cores. The hybrid inference model, which incorporates flow-level state computed on the cores, improves detection performance over purely stateless execution while preserving line-rate processing. Concurrently, the system maintains low and stable latency, with performance close to baseline L2 forwarding. This architectural design can be extended to other SmartNIC platforms with comparable fast-path and on-board cores.

Acknowledgements

This work was partly funded by the Leverhulme Trust and EU Horizon SmartEdge (101092908, Innovate UK 10056403).

References

[1] P. Bosshart et al. P4: programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.
[2] C. Zheng et al. In-network machine learning using programmable network devices: A survey. *IEEE Commun. Surv. Tutor.*, 26(2), 2024.
[3] R. Parizotto et al. Offloading machine learning to programmable data planes: A systematic survey. *ACM Comput. Surv.*, 56(1), August 2023.
[4] E. F. Kfoury et al. A comprehensive survey on SmartNICs: Architectures, development models, applications, and research directions. *IEEE Access*, 12:107297–107336, 2024.
[5] R. A. Bakar et al. Next-generation intrusion prevention system using hardware-accelerated data processing units (DPUs). In *Proc. of IEEE ICC (ICC Workshops)*, pp. 1438–1443, 2025.
[6] K. Tasdemir et al. An investigation of machine learning algorithms for high-bandwidth SQL injection detection utilising BlueField-3 DPU technology. In *Proc. of IEEE SOCC*, pp. 1–6, 2023.
[7] R. Kapoor et al. ML-NIC: accelerating machine learning inference using smart network interface cards. *Front. Comput. Sci.*, 6:1493399, 1 2024.
[8] Intel® Infrastructure Processing Unit. https://www.intel.com/content/www/us/en/products/details/network-io/ipu/adapter-e2100.html.
[9] A. T.-J. Akem and N. Zilberman. HyNIC: Hybrid in-network inference for line-rate anomaly detection on SmartNICs. In *IEEE NetSoft*, 2026.
[10] A. Alsaedi et al. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access*, 8:165130–165150, 2020.