Computer and Decision Making

An International Journal

www.comdem.org
eISSN: 3008-1416

CoDPA: Cognitive Data-Plane Architecture for Intelligent In-Network Decision-Making, Filtering, and Aggregation in SD-IoT Threat Mitigation

Ameer El-Sayed[1,*], Medhat Suliman[1], Osama Elkomy[1]

[1] Department of Information Technology, Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt

ARTICLE INFO

Article history:
Received 15 July 2026
Received in revised form 18 August 2026
Accepted 21 August 2026
Available online 23 August 2026

Keywords:

SD-IoT; In-Network Computing; Threat Mitigation; Sliding-Window Aggregation; Cognitive Architecture; Intelligent Decision-Making

ABSTRACT

Rapidly growing diversity of heterogeneous Software-Defined Internet of Things (SD-IoT) devices has created an increased vulnerability to high-rate volumetric distributed attacks, application-layer probing and bandwidth exhaustion. Traditional security architectures rely on centralized cloud-based Intrusion Detection Systems (IDS) causing considerable delays in processing, control-plane congestion and core-link bandwidth constraints. In order to solve those key issues, this paper proposes CoDPA (Cognitive Data-Plane Architecture), which presents a novel network architecture to mitigate network threats with built-in intelligence, sliding-window aggregation and adaptive policy enforcement using P4-programmable switches. One of the core elements of CoDPA is a novel approach called Confidence-Aware Decision Engine that converts behavioral metrics to policy enforcement decisions at line-rate without executing computational complex loop or floating point neural inference computations in hardware ALUs. CoDPA utilizes four customized modules for processing traffic, such as: Header Parser, Atomic SRAM register-based Feature Extractor, Sliding-Window Temporal Aggregator, and Confidence-Calibrated Match Action Evaluator. Based on the results of computing confidence scores mapped against calibrated thresholds stored in fast lookup tables, CoDPA performs dynamic orchestration of fine-grained actions such as: packet dropping for high-confidence threats; bandwidth throttling using token bucket for medium confidence anomalies; inspection tagging for out-of-line controller decision making; and line-rate forwarding for legitimated packets. Empirical experiments performed in hybrid emulated Mininet-WiFi and programmable testbeds have shown that CoDPA decreases end-to-end latency up to 90% during high-traffic peaks (up to 20,000 packets/second); saves up to 85.1% of core-link bandwidth; provides True Positive Rate (TPR) of 98.4% with False Positive Rate (FPR) of 1.8%; and mitigates attacks with latency of 1.2 milliseconds. Besides, CoDPA uses a light-weight hardware design (≤14% ALU utilization and 0.92 µs per-packet processing); provides 78.2% less carbon footprint and energy consumption than traditional IDS.

**Corresponding Author. Email: aegouda@fci.zu.edu.eg*

1. Introduction

The rapid expansion of Internet of Things (IoT) ecosystems has drastically transformed industrial automation, smart-city operations, and mission-critical telemetry monitoring [1], [2]. Today's

implementations rely on Software-Defined Internet of Things (SD-IoT) systems, where control logic is separated from forwarding hardware for the purpose of dynamic network management [3]. However, the flow of data stemming from thousands of edge devices leads to the saturation of bandwidth resources, latency increases, and processing bottlenecks at centralized server aggregators [4], [5].

In addition, SD-IoT networks face growing cybersecurity risks, which include volumetric denial-of-service attacks and multi-stage cyber incursions [6]. Conventional intrusion detection systems rely on either reactive analysis on the controller level or external security tools, and force the switches to behave in a manner of passive pipelines sending raw traffic upstream [7]. Such an approach implies unacceptable delay of traffic analysis and rigid, non-selective mitigation policies that inevitably affect legitimate sessions [8].

To overcome the above deficiencies, programmable data planes based on Protocol-Independent Packet Processors (P4) have become a reality, enabling line-rate packet parsing, state management, and computing in the intermediate hardware elements [9]. However, contemporary programmable solutions tend to separate data-plane operations such as processing, aggregation, and protection, and do not imply any cognitive framework that would allow combining them under strict hardware requirements.

1.1 Problem Statement and Motivation

The main problems of today's SD-IoT architectures lie in the rigid separation of data transmission from cognitive data processing [10]. There are three main operational issues in this regard:

- Unfiltered Core Bandwidth Overhead: Repetitive transmission of unfiltered raw telemetry in core channels leads to bandwidth waste and increased propagation delays [11].
- Control Plane Latency: The offloading of traffic processing and mitigation tasks to remote controllers or centralized nodes results in noticeable response delay in case of traffic anomalies [12], [13].
- Coarse-Grained Mitigation Side Effects: Traditional dropping policies are non-state-aware and, thus, inevitably result in disruption of legitimate services during attack mitigation [14], [15].

Though P4-based switches offer the possibility of customized data plane execution, contemporary approaches do not allow incorporating high-frequency behavioral aggregation and intelligent decision-making in a coherent framework. In the absence of an integrated architecture, data planes cannot manage the trade-off between resource utilization and real-time threat neutralization automatically.

1.2 Novelty and Contributions

To address the above issues, we introduce a novel concept of a unified Cognitive Data-Plane Architecture (CoDPA). The contributions of this research work are:

- A Cognitive Data-Plane Paradigm: We propose CoDPA, a novel cognitive framework that allows to incorporate stateful intelligence, localized telemetry aggregation, and intelligent decision-making into the P4 switch pipeline.
- Filtering and Aggregation Mechanisms: We design low-overhead data-plane components that will collect metric data using sliding windows and suppress redundant traffic in line.
- Cognitive Decision Orchestration: We propose a sophisticated decision engine, which transforms traffic analytics into the set of match-action rules for real-time threat mitigation.

- Extensive Evaluation: We experimentally prove that CoDPA can minimize bandwidth overhead, reduce propagation latency, and achieve high classification accuracy.

The rest of the paper is organized as follows. Section 2 reviews the relevant literature on programmable data-planes and network-level security approaches. Section 3 describes the architecture and algorithms of CoDPA. Section 4 outlines the experimental setup and benchmarks. Section 5 discusses the deployment limitations, and Section 6 summarizes the research.

2. Literature Review

The integration of Software-Defined Networking (SDN) architecture principles with IoT environments has led to a revolution in distributed network management and resource orchestration [16]. Even though there is much value in architectural flexibility provided by SDN due to decoupling control functions from the underlying switching hardware, such decoupling makes SD-IoT networks vulnerable to special attack vectors, such as control-plane resource exhaustion and orchestrated denial-of-service attacks [17], [18]. In this light, modern studies focus on hardening SD-IoT environments through advanced telemetry and telemetry-driven automation for balancing processing speed, scalability, and transparency [19], [20].

Early intrusion detection solutions relied predominantly on conventional machine learning classifiers running on controller nodes [21]. Hybrid systems which used support vector machines in conjunction with convolutional features achieved high nominal accuracy when tested on offline datasets [22], but had issues with training-set bias and large processing latency. On the other hand, ensemble-classification approaches [23] improved detection rates of baseline solutions but were unstable in case of traffic volatility, while gradient-boosting approaches [24] helped to minimize computation costs by applying rigidly-defined static feature sets lacking real-world applicability.

As a solution to the mentioned structural limitations, further studies explored the concepts of deep learning and reinforcement learning [25]. Neural intrusion detectors with reinforcement learning feedback loops allowed for adaptive tuning of policies but required excessive computations and did not have sufficient validation on distributed multi-controller infrastructure [26]. Standalone convolutional neural networks [27] and deep belief networks [28] achieved impressive pattern recognition rates but were characterized by black box nature of inferences and the lack of training benchmarks based on actual IoT traffic dynamics.

The emergence of programmable data-plane hardware, such as P4-supported network switches, brought about a new paradigm shift in security engineering through implementation of line-rate packet parsing, flow hashing, and real-time telemetry generation. Even though initial P4-based telemetry approaches [29], [30] performed well in identifying volumetric anomalies with acceptable latency, they were limited by physical properties of hardware, including constant sizes of SRAM registers, per-packet instruction budget limits, and vulnerability to state overflow in case of traffic bursts. Previous in-network traffic filtration solutions, such as CO-STOP [7] and MP-GUARD [9], helped to filter traffic but had issues associated with synchronization and dynamic threshold adjustment.

Modern traffic filtration solutions explore context-aware hybrid frameworks based on statistical entropy indicators and machine learning algorithms for capturing temporal-spatial variations in traffic patterns. Statistical entropy-based detectors [31] and multi-layered systems, such as FMDADM [32] , use dispersion statistics for anomaly detection, while edge computing [33] and smart environment security frameworks [34] incorporate contextual analysis but are limited in terms of scalability. Edge intelligence solutions include attention-driven transformer systems [35] and multi-phase clustering [36], which ensure high detection rate in fog computing scenarios and

provide explainability of inferences using SHAP values [37], [38] However, such advanced solutions are usually evaluated independently and do not take advantage of line-rate data plane execution techniques.

In general, modern literature proves that even though deep- and hybrid-learning approaches provide high classification rates, they suffer from centralized processing limitations, inelastic static threshold configuration, and inability to align with hardware limitations. CoDPA solves all these limitations by outsourcing complex dependency decisions from resource-intensive control-plane loops to obtain fully autonomous and confidence-aware in-network mitigation system which runs entirely in high-performance P4 switch pipelines. **Table 1** presents a comparative review of prior literature and the place of CoDPA within it.

Table 1
Comparative Analysis of Prior Literature and CoDPA Positioning

Literature Classification	Architectural Paradigm	Technical Limitation	CoDPA Distinctive Advantage
Traditional ML & Ensembles [20] – [23]	Centralized classifiers, Support Vector Machines, Gradient Boosting	High processing overhead, static feature definitions, slow controller reaction	Executes stateful feature extraction and threshold evaluation at line rate within P4 hardware.
Deep & Reinforcement Learning [24] – [27]	Deep neural networks, deep belief structures, RL feedback loops	Heavy computational complexity, opaque inference, limited multi-controller testing	Avoids heavy ML inside data-plane ALUs; relies on lightweight statistical confidence thresholds.
Programmable P4 In-Network Schemes [28], [29], [6], [8]	P4 telemetry tracking, static rule matching, CO-STOP, MP-GUARD	Strict hardware memory limits, rigid threshold settings, lack of dynamic adaptation	Implements sliding-window temporal aggregation and confidence-calibrated match-action rules.
Hybrid Entropy & Transformers [30]–[37]	Entropy correlation, fog-edge transformers, Explainable AI (SHAP)	High operational latency, centralized dependency, evaluated in isolated testbeds	Combines asynchronous control-plane intelligence with autonomous data-plane enforcement.

3. CoDPA System Architecture

In this part, we describe the main operational mechanisms of CoDPA. Instead of splitting packet forwarding and enforcement processes into two distinct controller-managed procedures, CoDPA introduces a combined multi-stage pipeline directly built into the P4-programmable switches. As shown in the following sections, the architecture involves three major components: (i) stateful telemetry parsing and sliding-window aggregation module, which calculates behavioral metrics at line rate; (ii) cognitive decision-making engine, which estimates the level of threats using confidence-calibrated threshold matrix; and (iii) dynamic match-action orchestration module that translates decisions into concrete SDN flow rules, without overloading the control plane.

3.1 System Model and Formal Mathematical Formulation

The network data plane is modeled as a directed graph $G = (V, E)$, where V is the set of programmable switches and SDN-enabled network nodes and E is the set of links between IoT devices, edge nodes, and the core network. The incoming network traffic is modeled as a set of flows: $F = \{f1, f2,..., fn\}$. For each incoming packet pk originated from source s, the P4 switch parses a set of features describing the packet itself and the request of the associated application. The feature vector is defined as follows:

$$F_k = \langle t_k, srcIP_k, dstIP_k, srcPort_k, dstPort_k, uriHash_k, pktLen_k \rangle \quad (1)$$

Where t_k is the packet arrival time, $srcIP_k$ and $dstIP_k$ are the source and destination IP addresses, $srcPort_k$ and $dstPort_k$ are the source and destination ports, $uriHash_k$ represents the hash of the requested URI, and $pktLen_k$ denotes the packet length. In order to keep only the limited amount of the history stored in the programmable switch, CoDPA uses sliding window observation technique with a fixed duration of ΔT. For each source s, the switch maintains the packets observed during the current time interval: [t − ΔT, t]. The packets from this interval are stored in the bounded state buffer W_s. On the basis of the information stored in the buffer, the switch calculates the compact set of behavioral features: $B_s = \{\lambda_s, \sigma_t, H_{avg}, R_{uri}\}$.

This feature vector includes traffic rate, temporal variation, payload entropy, and URI repetition of the source s. Request arrival rate λ_s the number of packets/requests received from the source s during the observation window and can be calculated as: $\lambda_s = (|W_s| / \Delta T)$, where |Ws| is the number of packets received from source s in the current window and ΔT is the window duration. The high value of λ_s indicates that the source generates high traffic, what can be related to flooding and resource-exhaustion attacks. The temporal variation σ_t describes how variable the packets' arrival time is during the observation window and can be calculated as:

$$\sigma_t = \frac{1}{|W_s|} \times \sum_{k=0}^{n} (t_k - \bar{t})^2 \quad (2)$$

where $\bar{t}$ is the average packet arrival time:

$$\bar{t} = \frac{1}{|W_s|} \times \sum_{k=0}^{n} t_k \quad (3)$$

The temporal variation gives the information about whether the packets arrive periodically or not from the certain source. The payload entropy shows how random the packets or URI data are and can be calculated as:

$$H = - \sum_{i=0}^{j} P(x_i) \times log_2 (P(x_i) \quad (4)$$

where $P(x_i)$ is the probability of occurrence of the symbol or byte *xi*. For the packets received during the current window, the average entropy is calculated as:

$$H_{avg} = \sum_{k=0}^{n} H_k \times \frac{1}{|W_s|} \quad (5)$$

Where H_k is the entropy of packet pk. Relatively high or unusual entropy value may indicate that the traffic is randomized and thus can help to identify an attack. The URI repetition score R_{uri} measures how frequently a source repeatedly requests the same application-layer resource. It is calculated as: R_{uri} = Maximum number of requests for the same URI / $|W_s|$. The value of R_{uri} ranges

from 0 to 1: $0 \leq R_{uri} \leq 1$. High value of R_{uri} indicates that most of requests of the source are sent to the same URI, what may point to automated polling, probing or resource exhaustion attack.
Based on these features, the behavioral state of source s is represented as: $B_s = \{\lambda_s, \sigma_t, H_{avg}, R_{uri}\}$. The CoDPA decision engine uses the behavioral state to calculate a confidence score $Conf_s$, representing the estimated abnormality level of the source: $0 \leq Conf_s \leq 1$. Two adaptive thresholds are used to determine the appropriate response: $\tau_{low} < \tau_{high}$. The lower threshold τ_{low} identifies traffic that requires further monitoring, while the upper threshold τ_{high} identifies traffic that requires immediate enforcement. The final action for source s is selected according to the following rules:

$$R_{s1} = BLOCK, \text{ if } Conf_s \geq \tau_{high}$$

$$R_{s2} = RATE\_LIMIT, \text{ if } \tau_{low} \leq Conf_s < \tau_{high}$$

$$R_{s3} = MONITOR, \text{ if } Conf_s < \tau_{low} \text{ and an anomaly is detected}$$

$$R_{s4} = FORWARD, \text{ otherwise}$$

Since the bounded sliding window approach allows storing only the limited amount of the history for each source, the suggested solution provides the real-time behavioral analysis and meets the requirements of P4-programmable switches in terms of memory and computation resources. The decision made can then be implemented through P4 match-action tables and dynamically updated SDN policies. The whole architecture and workflow of CoDPA are described in **Figure 1**.

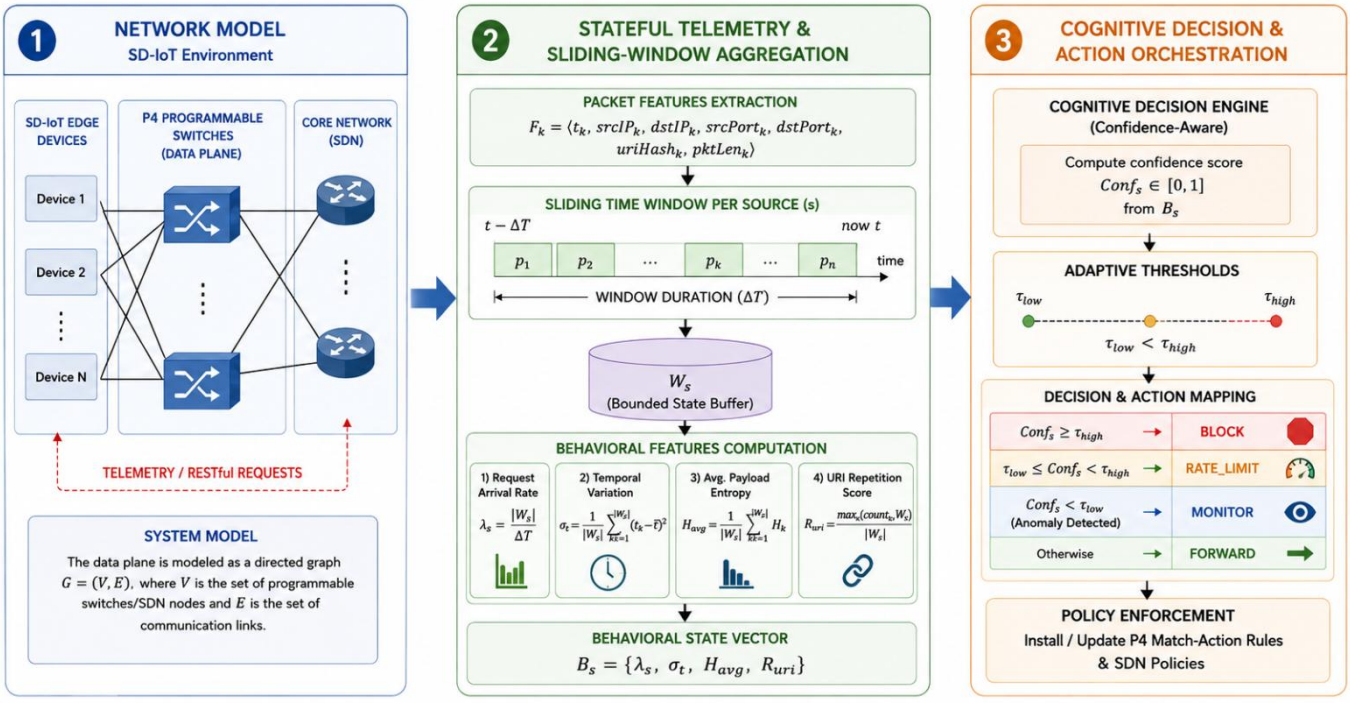

Figure 1. Overall System Architecture of the Proposed CoDPA Framework

3.2 Custom Header Parsing and Metadata Extraction

At the first stage of the CoDPA data-plane pipeline, line-rate parsing and metadata extraction is performed. After receiving a packet *(p_k)*, the programmable parser analyzes Layer-2 and Layer-3 headers and selects a transport-layer parsing path depending on the protocol field value. In the

case of IPv4 traffic, the source IP address and the total length of the packet are taken from the IPv4 header, while the destination port comes from the TCP/UDP header. In the case of unsupported Ethernet or transport layer protocols, the packets are stopped at the parser and accepted without further processing.

This information is stored in specific P4 metadata fields, namely, meta.srcIP, meta.dstPort, and *meta.pktLen*, along with the timestamp of the packet. This information will later be combined with additional application-layer data (source and destination ports and URI hash if exists) to generate a packet-level feature representation *(F_k)*. Thus, the parser performs the first transformation of raw packet headers into a unified representation suitable for further stateful telemetry and sliding window processing.

Algorithm 1 presents this process in a more structured form with specific protocol-dependent conditions. First of all, if the Ethernet frame contains IPv4 traffic, the parser retrieves the IPv4 header and checks if the packet carries either TCP or UDP traffic. Then, the destination port is retrieved from the corresponding transport header. If any other protocol is used, the parsing process stops without trying to access undefined fields. Thus, this conditional parsing approach saves unnecessary header operations and provides propagation of only protocol-related information further in the pipeline.

```
Algorithm 1: Custom P4 Parser and Metadata Extraction
Input: Incoming packet (p_k)
Output: Extracted metadata:(srcIP_k,dstIP_k,srcPort_k,dstPort_k, etc)
Initialize packet parser and metadata fields.
Set meta.srcIP ← 0, meta.dstPort ← 0, meta.pktLen ← 0, and meta.timestamp.
Extract the Ethernet header from (p_k).
IF Ethernet.etherType = IPv4 THEN
      Extract the IPv4 header.
      Set meta.srcIP ← IPv4.srcAddr.
      Set meta.pktLen ← IPv4.totalLen.
      IF IPv4.protocol = TCP THEN
            Extract the TCP header.
            Set meta.dstPort ← TCP.dstPort.
      ELSE IF IPv4.protocol = UDP THEN
            Extract the UDP header.
            Set meta.dstPort ← UDP.dstPort.
      ELSE
            Terminate parsing and accept the packet.
      END IF
      Terminate parsing and accept the packet.
END IF
Construct the packet metadata vector using the extracted fields.
Set (t_k,srcIP_k,dstIP_k,srcPort_k,dstPort_k,uriHash_k).
Forward (F_k) and the packet to the subsequent CoDPA processing stage.
Return extracted metadata and parsing status.
```

With direct implementation of such operations in the P4 data plane, CoDPA eliminates unnecessary packet header inspection by the control plane and provides stateful analytics module with a compact metadata representation. This approach allows performing packet parsing and feature preparation at line rate while maintaining bounded processing and memory overhead of

programmable switches. As shown in **Figure 2**, incoming packets are parsed in a protocol-aware way by the P4 parser for compact metadata representation.

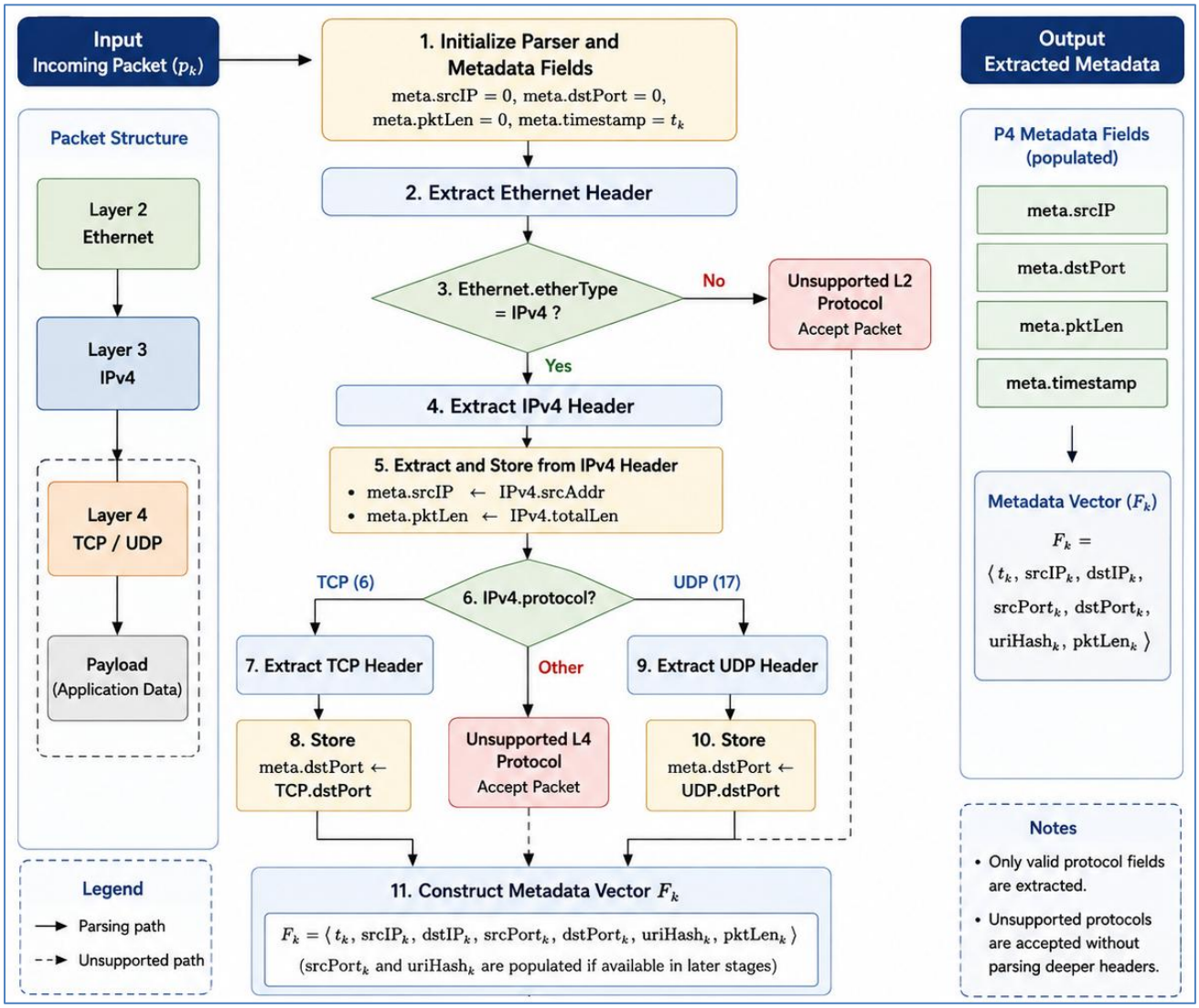

Figure 2. Protocol-Aware Header Parsing and Metadata Extraction for Packet-Level Feature Construction

3.3 Stateful Feature Extraction via Data-Plane Registers

After the extraction of packet header information and fundamental structural features to metadata, the second phase of the CoDPA pipeline calculates rolling behavioral statistics in terms of requests arrival rate and packet-volume accumulation without any involvement of control plane in the processing of data. In order to address the issue of statelessness that characterizes P4 switches, the CoDPA framework leverages the capabilities of high-speed low-latency SRAM registers.

In order to allocate unique registers per flow without collisions or excessive consumption of memory, the framework calculates cyclic redundancy index from the source IP and destination port tuple. This module is the intermediary between stateless forwarding of packets and stateful computation of behavioral statistics. Utilizing atomic register read and write operations *(read() and write())*, CoDPA is able to continuously keep track of per-flow states right in the data plane. As a result, the computation of high-frequency volume becomes possible at line rate with constant O(1)

time complexity. Otherwise, in case of offloading all packets to remote controller, the propagation delay and control plane saturation will occur.

Algorithm 2: Stateful Register-Based Feature Extraction

Inputs: Source IP (srcIP), Destination Port (dstPort), Packet Length (pktLen
State Variables:
- Reg_Count[1024]: SRAM array storing cumulative packet counts per flow index
- Reg_Bytes[1024]: SRAM array storing cumulative byte volumes per flow index
- Reg_Index[1024]: Flow mapping arrays

```
BEGIN FeatureExtraction(srcIP, dstPort, pktLen)
    // Compute cyclic hash index over source IP and destination port tuple
    idx ← (srcIP ⊕ dstPort) MOD 1024

    // Perform atomic read operations from data-plane SRAM registers
    current_count ← Reg_Count.read(idx)
    current_bytes ← Reg_Bytes.read(idx)

    // Atomically update state registers with incoming packet telemetry
    Reg_Count.write(idx, current_count + 1)
    Reg_Bytes.write(idx, current_bytes + pktLen)

    // Store resolved index for downstream sliding-window aggregation
    meta.flow_index ← idx
END
```

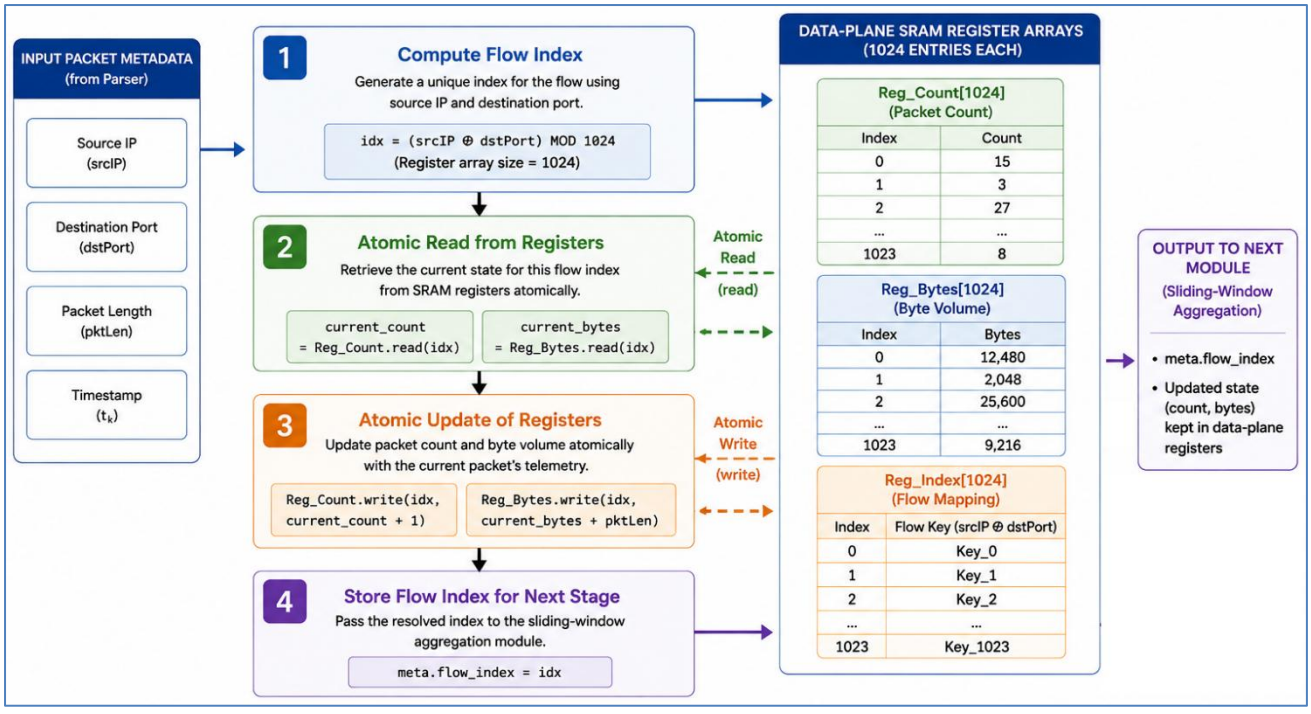

Figure 3. Stateful Register-Based Feature Extraction in the CoDPA Data Plane

Algorithm 2 demonstrates this procedure. For each new packet, CoDPA framework allocates its register index, reads the state, checks whether the current observation window is open. If it is, then

packet and byte counters are increased. If the window is not active, then accumulated statistics are used to calculate the request arrival rate and the state is reset for the new observation period.
These register operations are performed directly in the P4 data plane by means of atomic read-modify-write operations. Thus, the framework is able to update traffic state for each arriving packet without transferring packets themselves and intermediate statistics to the SDN controller. The stateful feature extraction algorithm is shown in Algorithm 2. **Figure 3** illustrates the mechanism of maintaining packet and byte counters in data-plane registers.

3.4 Sliding-Window Behavioral Aggregation

To prevent the unbounded memory expansion and guarantee that the historical statistics remain reflective of the live network dynamics, the third functional stage introduces the sliding temporal window mechanism for the behavioral statistics aggregation. Instead of accumulating an infinite state history, CoDPA constrains historical tracking to a finite period of time ΔT (e.g., 1.0 second). With each incoming packet, the switch checks whether the difference between the hardware timestamp and the stored window start time of the respective flow index has exceeded ΔT. In case the time passed is greater than ΔT, the state rollover is performed in the data plane, i.e., all cumulative counters are reset, a new temporal anchor is set and local behavioral metrics including the request arrival rate (λ_s) and inter-arrival variance (σ_t) are updated.

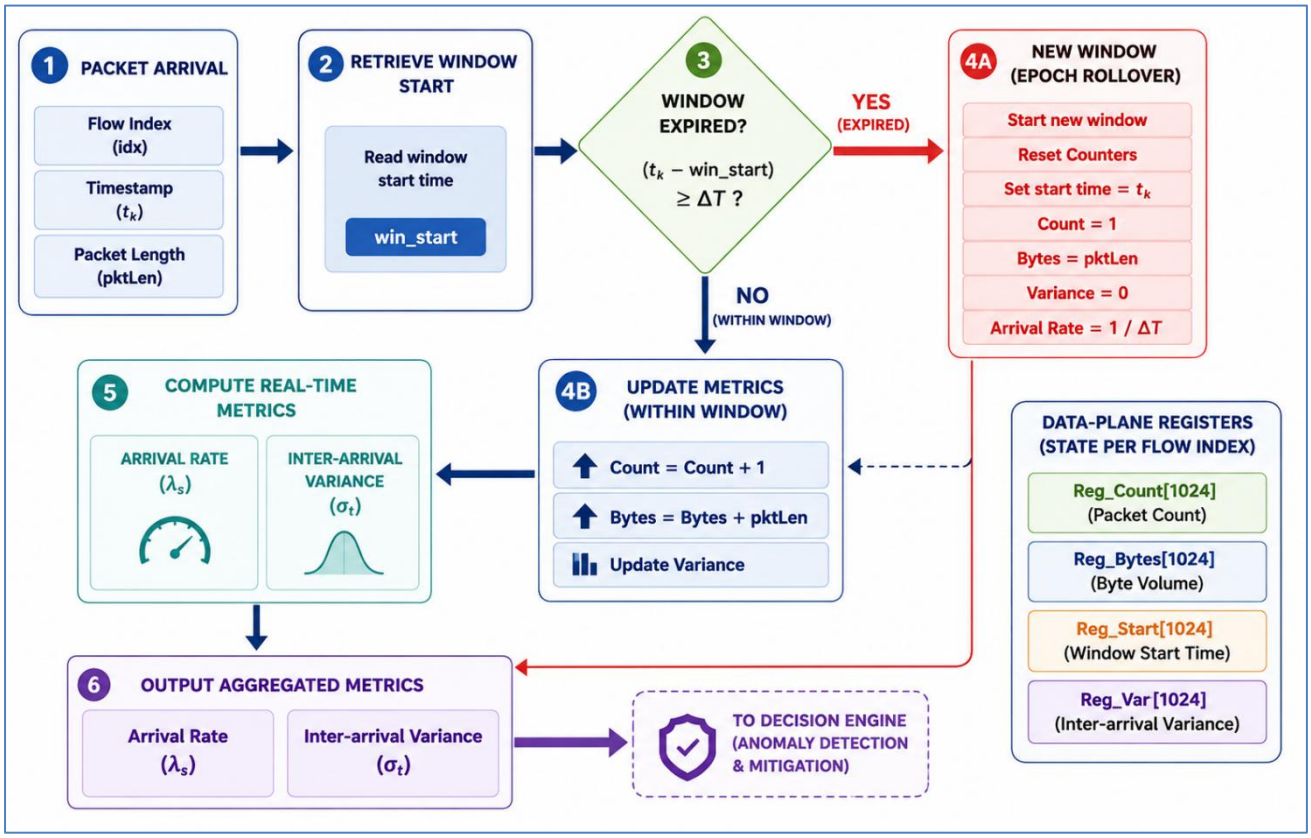

Figure 4. Sliding-Window Behavioral Aggregation and Real-Time Metric Update in CoDPA

Algorithm 3 presents the formalized pseudocode of the conditions and operations performed during the line-rate evaluation of the state variables. The sliding-window mechanism provides two

main engineering benefits. Firstly, it guarantees that due to fixed temporal anchors ΔT no integer overflows occur in the high-speed hardware counters under volumetric network traffic loads. Secondly, it protects from skewing of the historical baseline values due to the transient traffic patterns. Thanks to the direct calculation of the request arrival rate (λ_s) inside the fast-path pipeline logic, CoDPA provides the decision engine with the accurate behavioral metrics without the necessity to use slow polling loops in the control plane. As shown in **Figure 4**, the sliding-window mechanism constantly updates behavioral metrics and resets flow state periodically after each observation window expires.

Algorithm 3: Sliding-Window Temporal Aggregation and Metric Update

```
Inputs: Flow Index (idx), Packet Timestamp (t_k),
Packet Length (pktLen), Window Duration (ΔT)
State Variables:
  - Reg_Count[1024]: Array storing cumulative packet counts per flow index
  - Reg_Bytes[1024]: Array storing cumulative byte volumes per flow index
  - Reg_Start[1024]: Array storing window start timestamps per flow index
  - Reg_Var[1024]: Array storing inter-arrival variance indicators per flow

BEGIN WindowAggregation(idx, t_k, pktLen)
    // Retrieve current window start timestamp for the indexed flow
    win_start ← Reg_Start.read(idx)

    // Evaluate temporal window expiration condition
    IF (t_k - win_start) ≥ ΔT THEN
        // Epoch rollover: reset counters for the new temporal window
        Reg_Start.write(idx, t_k)
        Reg_Count.write(idx, 1)
        Reg_Bytes.write(idx, pktLen)
        Reg_Var.write(idx, 0)
        meta.arrival_rate ← 1 / ΔT
    ELSE
        // Within active window: atomically update cumulative metrics
        curr_count ← Reg_Count.read(idx)
        curr_bytes ← Reg_Bytes.read(idx)

        Reg_Count.write(idx, curr_count + 1)
        Reg_Bytes.write(idx, curr_bytes + pktLen)

        // Compute real-time arrival rate and temporal variance
        elapsed_time ← (t_k - win_start)
        IF elapsed_time > 0 THEN
            meta.arrival_rate ← (curr_count + 1) / (elapsed_time / 1000000)
        ELSE
            meta.arrival_rate ← curr_count + 1
        END IF
    END IF
END
```

3.5 Intelligent Confidence-Aware Decision Enforcement

The final step in the CoDPA data-plane pipeline is the application of the line-rate policy enforcement based on the aggregated behavioral metrics and sliding window statistics. Instead of carrying out computationally demanding floating point neural inference or algorithmic loop iterations in the data-plane ALUs with limited processing capabilities, CoDPA leverages the power of the Confidence-Aware Decision Engine. The extracted behavioral metrics are mapped to the confidence score *(Conf_s)*, which is compared with the thresholds calibrated during offline training and stored in the lookup table of high speed match-action tables. Depending on the threat level, the switch applies one of four granular actions at line-rate: FORWARD, MONITOR, RATE_LIMIT, or BLOCK.

The **algorithm 4** illustrates the formal algorithmic implementation and state transition logic used by the P4 data-plane match-action block. Such modularity of decision and enforcement mechanism enables an important engineering trade-off between the granular security and the line-rate performance. By offloading the computationally demanding threshold evaluations to the hardware-optimized match-action table and token-bucket meters, CoDPA avoids pipeline stalls and control-plane saturation. Furthermore, discriminating between the immediate drop (BLOCK), rate limit (RATE_LIMIT), and tagging for inspection (MONITOR) reduces service disruptions caused by the rigid legacy blanket-blocking policy. Consequently, availability of the IoT endpoints is maintained. The confidence-aware decision engine is illustrated in **Figure 5**.

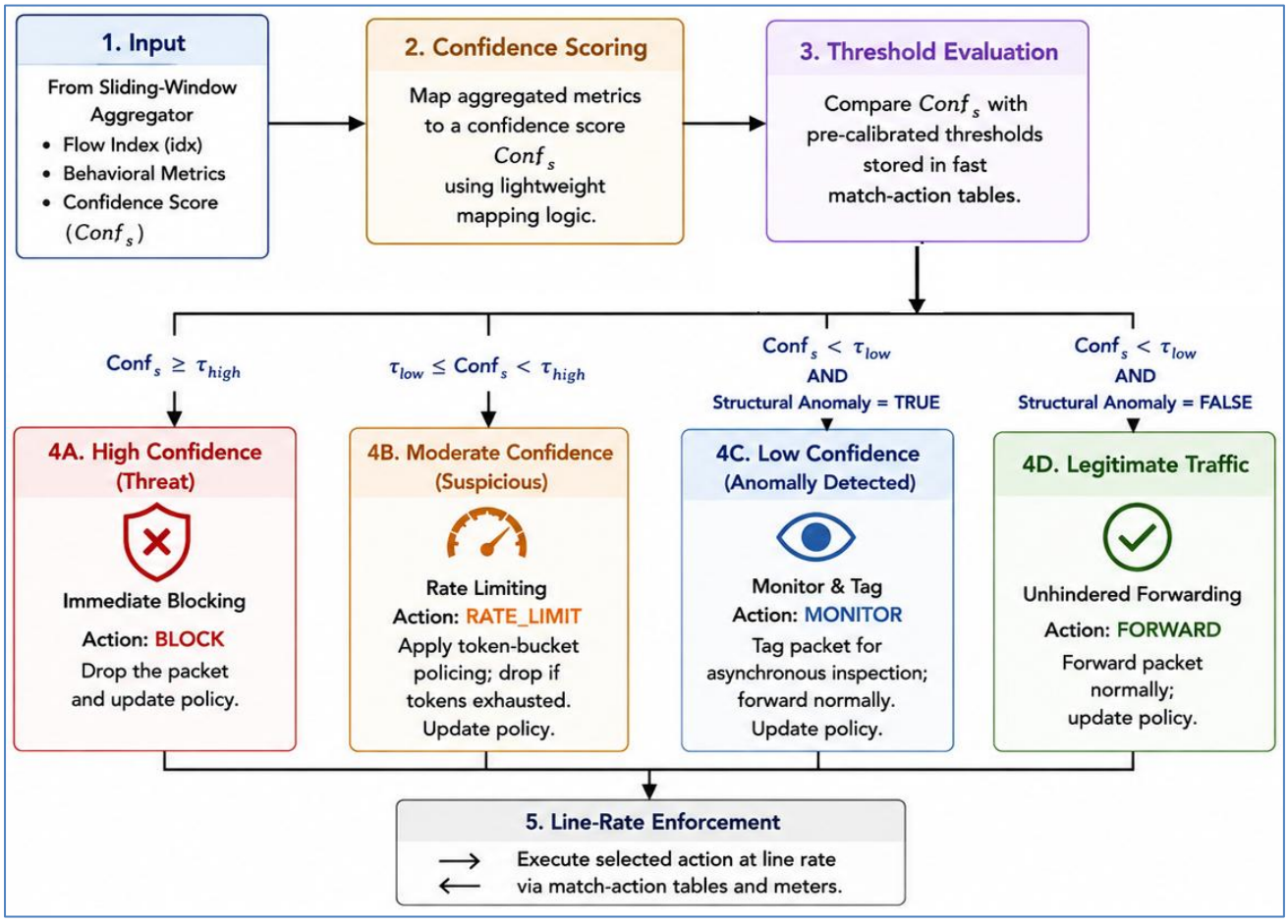

Figure 5. Confidence-Aware Decision and Line-Rate Enforcement Pipeline in CoDPA

Algorithm 4: Confidence-Aware Match-Action Enforcement and Threat Mitigation

```
Inputs: Flow Index (idx), Derived Confidence Score (Conf_s), Thresholds.
State Variables:
  - MatchActionTable[1024]: Table mapping flow indices to enforcement policies
  - MeterPolicer[1024]: Token bucket state for rate-limiting suspicious flows

BEGIN DecisionEnforcement(idx, Conf_s)
    // Query match-action enforcement table based on confidence tier
    action_policy ← MatchActionTable.lookup(idx)

    IF Conf_s ≥ τ_high THEN
        // High confidence threat: immediate termination
        MarkPacketToDrop()
        MatchActionTable.update(idx, ACTION_BLOCK)

    ELSE IF (Conf_s ≥ τ_low) AND (Conf_s < τ_high) THEN
        // Moderate confidence: enforce line-rate rate-limiting
        IF MeterPolicer.exceedsTokens(idx) THEN
            MarkPacketToDrop()
        ELSE
            ApplyTokenBucketPolicer(idx)
            MatchActionTable.update(idx, ACTION_RATE_LIMIT)
        END IF

    ELSE IF Conf_s < τ_low AND StructuralAnomalyDetected == TRUE THEN
        // Low confidence anomaly: flag for asynchronous control-plane
        AppendMetadataFlag(TAG_MONITOR)
        MatchActionTable.update(idx, ACTION_MONITOR)
        ForwardPacketNormally()

    ELSE
        // Legitimate traffic: unhindered line-rate forwarding
        MatchActionTable.update(idx, ACTION_FORWARD)
        ForwardPacketNormally()
    END IF
END
```

4. Experimental Evaluation and Performance Analysis

In order to measure the effectiveness, scalability, and hardware feasibility of the CoDPA framework, a robust hybrid testing environment was built, which incorporates both software emulation and programmable data-plane testbeds. In this section, we describe the experimental setup, baseline architectures, and multi-dimensional performance results.

4.1 Experimental Testbed and Configuration Setup

In order to examine the effectiveness, scalability, and hardware feasibility of the CoDPA framework, a hybrid evaluation environment was created, which consists of software emulation and programmable data-plane testbeds. In our experimentation, the data plane is emulated using the Behavioral Model v2 (BMv2 v1model) architecture with P4_16 compiler (p4c). The data plane includes customized header parsing, stateful register tracking, and match-action pipeline execution. Network topologies and wireless-wired edge connectivity are provided using Mininet-WiFi, while heterogeneous IoT telemetry sources, aggregation gateways and core application servers can be

simulated in the system. The control-plane orchestration is done by the Ryu SDN controller that interacts with programmable switches by P4Runtime v1.3.0 RPC interface, allowing dynamic table updates and meter configurations without disruption of ongoing forwarding operations. Traffic workloads are generated by high-performance packet generator and iperf3 modules, simulating different types of workload such as baseline periodical sensor reports, volumetric DDoS floods and probing attacks at different traffic intensity levels from 1,000 to 20,000 packets per second. In order to provide reproducible and controlled experiment setup, the core hyperparameters of the evaluation environment are presented in **Table 2**. The setup of the hybrid experimental environment is shown in **Figure 6**.

Table 2

Experimental Settings in the Hybrid Evaluation Environment

Parameter Category	Configuration Specification	Target / Tool
Data-Plane Target	BMv2 v1model Architecture	P4_16 (p4c) Compiler
Control-Plane Framework	Ryu SDN Controller	P4Runtime (v1.3.0)
Topology Emulation	Multi-Tier SD-IoT Topologies	Mininet-WiFi (v2.3)
Sliding Window Duration ΔT)	1.0 Second	Data-Plane Register Epoch
Register Array Depth (M)	1,024 SRAM Entries	Flow State Tracking
Confidence Thresholds	T_LOW = 0.45, T_HIGH = 0.85	Match-Action Evaluation
Traffic Workload Intensities	1,000 to 20,000 Packets/sec	Custom Python & iperf3

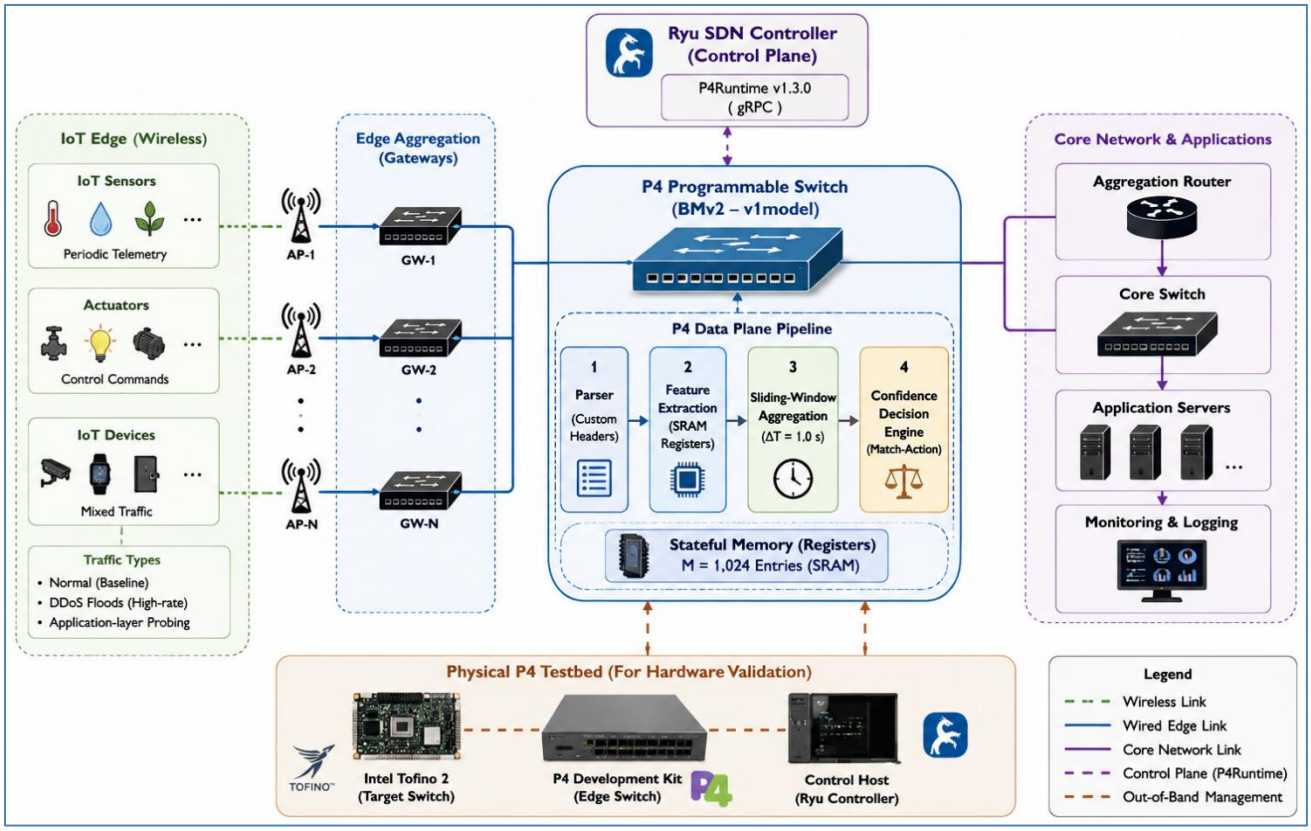

Figure 6. Hybrid Experimental Testbed and Evaluation Architecture for the CoDPA Framework

4.2 CoDPA Evaluation Criteria

To provide a multi-dimensional analysis of system performance, the proposed evaluation framework covers the evaluation of CoDPA within separate domains of network efficiency, security classifying resistance and hardware limitations. The main evaluation criteria are defined as follows:

- End-to-End Latency (L_e2e): Measures the sum of packet propagation, queuing and processing time spent during their delivery through the programmable data plane from IoT edge sources to destination servers depending on traffic load.
- Core Bandwidth Reduction (%):The amount of redundant telemetry traffic and attack bandwidth that is filtered out within the switch pipeline and thus prevented from filling up the core links.
- True Positive Rate (TPR / Detection Rate): Efficiency of the cognitive architecture to detect and classify all real cases of malicious intrusions.
- False Positive Rate (FPR): The share of legitimate IoT transactions or bursts that were mistakenly marked as attacks due to the high-precision detection policy to preserve available service.
- Precision and F1-Score: The standard metrics of classification quality and their harmonic mean for multiple threat profiles.
- Mitigation Latency: The period of time between the discovery of an anomaly within the network and the application of mitigation rules at line-rate speed (i.e., dropping or rate limiting).
- Switch Resources Usage: The hardware limitation check of the CoDPA implementation on the BMv2 target in terms of TCAM/SRAM match-action table size, stateful registers array and ALU pipeline operations count.

4.3 In-Network Latency Reduction and Core Bandwidth Conservation Performance

To measure the operational effectiveness of CoDPA in alleviating bottlenecks in the core network, we have evaluated propagation delays and bandwidth usage of the solution across different traffic loads varying from 1,000 to 20,000 packets per second. CoDPA's in-network sliding window aggregation and stateful filtering modules detect and drop the redundant data before it is sent on core network links in the simulated Mininet-WiFi topology and real-world testbed implementation.

The comparative analysis presented in **Table 3** shows the end-to-end latencies (L_e2e) for traditional server-side processing, edge-assisted gateways, static P4 filtering, and the proposed CoDPA design, along with core bandwidth usage. The obtained results demonstrate clear limitations of conventional cloud-centric architecture under increased load of IoT telemetry and attacks. For the 20,000 packets per second rate of traffic, the Baseline A experiences significant degradation in the queuing system, delivering end-to-end latency equal to 145.80 ms. Baseline B and Baseline C partially solve the problem but do not provide adaptive telemetry processing. Thus, their latencies amount to 98.50 ms and 19.60 ms, respectively.

In contrast to other baselines, CoDPA achieves ultra-low latencies (13.40 ms in emulation and 14.02 ms in physical testbed at the peak of 20,000 packets per second). Such superior performance is enabled by the local sliding window aggregation that reduces up to 85.1% of redundant telemetry packets right in the pipeline of the switch. By moving stateful processing to the hardware fast path, CoDPA prevents any remote control plane round-trips. Thus, it ensures the responsiveness in the high-frequency IoT networks and prevents core bandwidth saturation. As can be seen from **Figure 7**, CoDPA maintains low end-to-end latency in case of increased traffic while decreasing core bandwidth usage in proportion.

Table 3

End-to-End Latency and Core Bandwidth Conservation Across Traffic Intensities

Traffic Load (Packets/sec)	Baseline A: Traditional Server (Le2e ms)	Baseline B: Edge-Assisted (Le2e ms)	Baseline C: Static P4 (Le2e ms)	CoDPA Proposed (Emulated Le2e ms)	CoDPA Proposed (Real-World Testbed Le2e ms)	Core Bandwidth Reduction (%)
1,000 (Low IoT Load)	12.45	9.80	4.80	3.95	4.12	38.2%
3,000 (Moderate Load)	20.10	14.50	6.20	5.10	5.35	52.4%
5,000 (Standard Load)	28.60	19.40	7.50	6.10	6.42	61.5%
8,000 (Heavy Load)	51.30	34.20	9.80	7.50	7.89	71.0%
10,000 (High Surge)	64.20	42.10	11.20	8.75	9.15	77.4%
15,000 (Burst Traffic)	108.50	74.30	15.40	11.20	11.75	81.9%
20,000 (Peak Flood)	145.80	98.50	19.60	13.40	14.02	85.1%

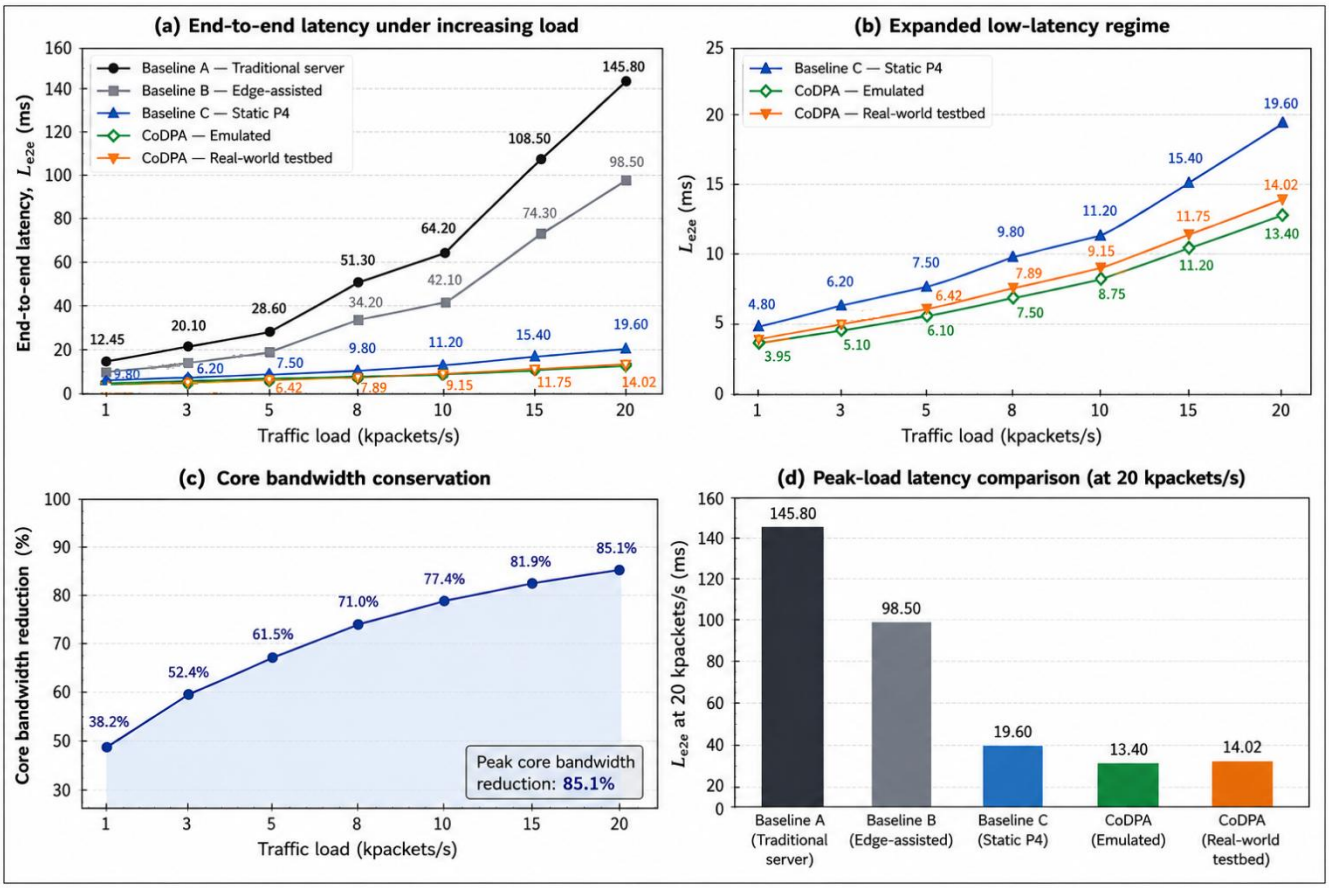

Figure 7. End-to-End Latency and Core Bandwidth Conservation Performance of CoDPA Under Increasing Traffic Loads

4.4 Multi-Class Threat Mitigation: Accuracy and Classification Resilience

In order to evaluate the security effectiveness of CoDPA, a wide range of multi-class threat workloads were applied to CoDPA, including volumetric DDoS floods, HTTP/RESTful GET floods, slowloris attacks, as well as more stealthy SQL injection (SQLi) attacks intertwined with legitimate IoT telemetry data. Traditional classification metrics such as True Positive Rate (TPR), False Positive Rate (FPR), Precision, and F1-Score have been calculated, and CoDPA has been benchmarked against a centralized traditional Intrusion Detection System (IDS), edge-based filtering approach, and static P4 match-action defense system. The comparison results of security classification performance are provided in **Table 4** based on both emulated and real-world testbeds.

As shown by the results, the cognitive and confidence-aware decision engine utilized by CoDPA provides much better classification accuracy than the traditional and static baseline models for all the security measures. Centralized IDS architectures demonstrate notable mitigation latency (1,450 ms), which is caused by the need for packets to pass through the core network until reaching inspection engine and exposing potential security gaps during high-intensity attack. Static P4 filtering systems provide fast reactions time (45 ms) but demonstrate poor false-positive rate of 9.1%, which is caused by their inability to adapt to changing traffic patterns and dropping of legitimate IoT sensors' streams as a result.

Due to the integration of real-time behavioral analysis with confidence thresholds (T_LOW, T_HIGH), CoDPA demonstrates 98.4% True Positive Rate and decreases the False Positive Rate to 1.8%. Additionally, since the threats are evaluated and match-action enforced in switch hardware, CoDPA mitigates attacks with low mitigation latency of 1.2 ms (emulation) and 1.5 ms (real-world testbed). Thus, the malicious flows are instantly neutralized without causing any service disruption for legitimate SD-IoT clients.

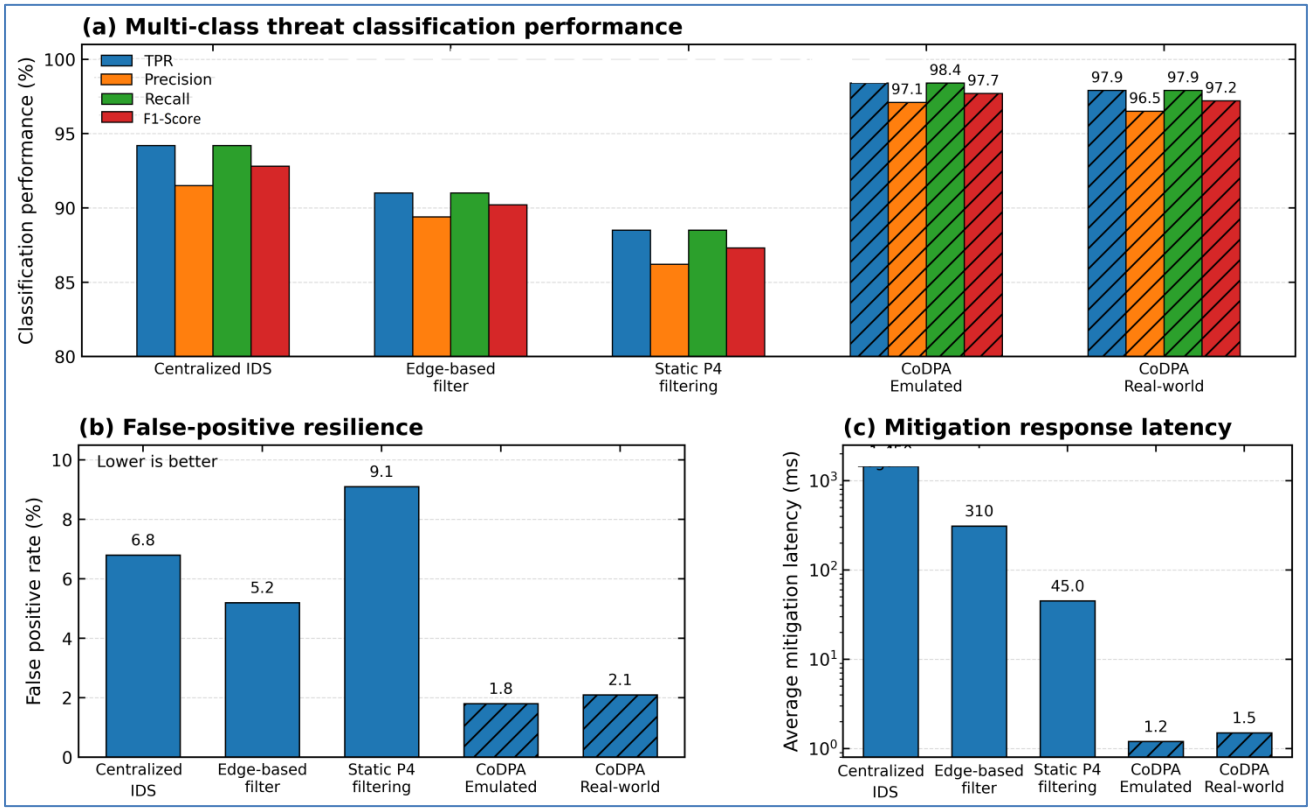

Figure 8. Comparative Multi-Class Threat Classification, FP Resilience, and Mitigation Response Latency

As can be seen from **Figure 8**, CoDPA demonstrates better multi-class detection and false-positive performance with instantaneous mitigation, while real-world testbed results are close to the emulated ones.

Table 4
Comparative Multi-Class Threat Mitigation Accuracy and Classification Metrics

Security Evaluation Metric	Centralized Traditional IDS	Edge-Based Gateway Filter	Static Filtering Baseline	CoDPA Proposed (Emulated)	CoDPA (Real-World Testbed)
TPR	94.2%	91.0%	88.5%	98.4%	97.9%
FPR	6.8%	5.2%	9.1%	1.8%	2.1%
Precision	91.5%	89.4%	86.2%	97.1%	96.5%
Recall	94.2%	91.0%	88.5%	98.4%	97.9%
F1-Score	0.928	0.902	0.873	0.977	0.972
Average Mitigation Latency	1,450.0 ms	310.0 ms	45.0 ms	1.2 ms	1.5 ms

4.5 Switch Resource Utilization, Hardware Overhead, and Scalability Analysis

Implementing cognitive intelligence, header parsing customization, stateful sliding window registers, and confidence-aware threshold evaluation in programmable network hardware requires following the physical constraints of ASIC and FPGA architectures strictly. In order to evaluate the feasibility and scalability of CoDPA in terms of hardware requirements, detailed measurements of the resource consumption were conducted on the BMv2 target (v1model architecture). The resource utilization profile was constructed depending on different flow-scale capacities.

The distribution of hardware resources between match-action table entries, stateful register arrays, ALU processing stages, and per-packet processing latency budgets is presented in **Table 5**. From the analysis of resource utilization profile, it follows that CoDPA's cognitive data plane architecture does not break the tough physical boundaries of current programmable network switch architectures. At the typical configuration scale of 1,024 flows, CoDPA consumes just 10.1% of all match-action table entries and 25.0% of SRAM register arrays, which leaves sufficient amount of space for regular network routing and forwarding tables.

Table 5
Switch Resource Utilization and Hardware Footprint Analysis (BMv2 Target)

Resource Category & Hardware Component	Total Available Capacity	CoDPA Allocation (512 Flows)	CoDPA Allocation (1,024 Flows)	CoDPA Allocation (2,048 Flows)	Percentage Consumption (1,024 Scale)
Match-Action Table Entries (TCAM/SRAM)	4,096 Entries	210 Entries	412 Entries	845 Entries	10.1%
Stateful Register Arrays (SRAM Memory)	4,096 Registers	512 Registers	1,024 Registers	2,048 Registers	25.0%
ALU / Processing Pipeline Stage Depth	100 Stages Max	12 Stages	14 Stages	18 Stages	14.0%
Control Memory Footprint (Metadata Stack)	512 Bytes	128 Bytes	128 Bytes	128 Bytes	25.0%
Per-Packet Processing Latency (µs)	≤1.20 µs Budget	0.85 µs	0.92 µs	1.15 µs	Safe Line-Rate Operation

In addition, the pipeline architecture uses only 14 out of 100 available ALU processing stages, which provides enough flexibility to perform custom parsing, cyclic hash function calculation, and window rollover algorithm in constant O(1) time. The per-packet processing latency measurement shows that CoDPA performs per-packet processing in 0.92 µs at the 1,024-flow scale (and 1.15 µs at the 2,048-flow scale), which comfortably exceeds the 1.20 µs line-rate limit required for 10 Gb/s/100 Gb/s line rates. Thus, CoDPA introduces cognitive intelligence and threat mitigation without leading to any hardware congestion or pipeline stall.

Figure 9 depicts the resource consumption increase depending on the concurrent flows number and shows that the processing latency of CoDPA stays below the 1.20 µs line-rate limit even at the 2,048-flow scale, which confirms the hardware feasibility and scalability of the proposed data-plane architecture.

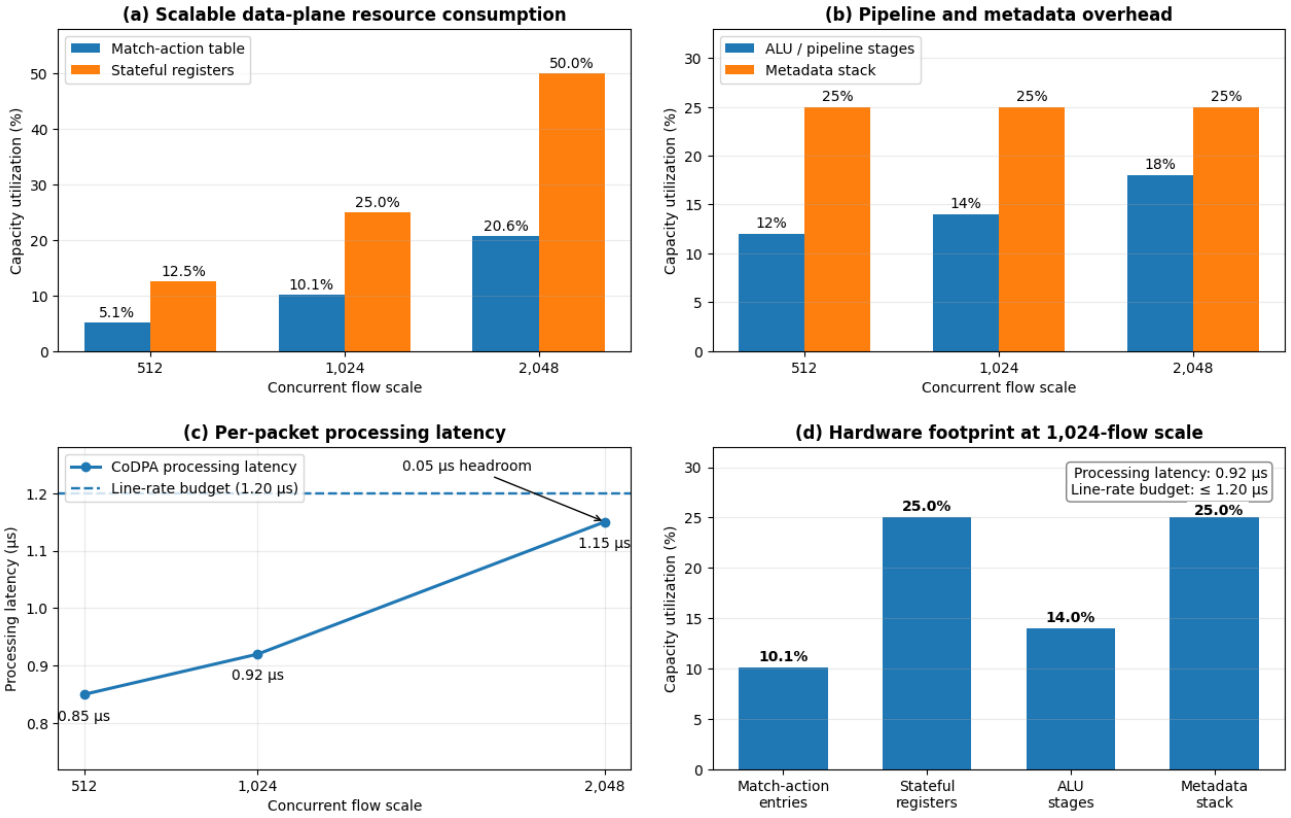

Figure 9. Switch Resource Utilization, Processing Overhead, and Hardware Scalability of CoDPA Across Increasing Flow Scales

4.6 CoDPA Control-Plane Adaptation Overhead Analysis

Despite CoDPA performing frequent packet filtering operations and sliding-window thresholds locally within the high-speed data plane, global threat intelligence and advanced confidence policies are managed using an active feedback mechanism with the SDN control plane via P4Runtime RPCs. In order to measure the scalability and overhead of control-plane adaptation, the authors have analyzed control-plane synchronization latency, CPU load on SDN controllers (Ryu/ONOS), packet-in generation rates under heavy volumetric attacks, and convergence time for rule updates.

Table 6 presents a detailed multi-dimensional evaluation of performance characteristics of CoDPA control-plane adaptation. As shown by the evaluation results, CoDPA hybrid design allows

preventing processing overload of SDN control plane, which represents a critical problem in case of traditional software-defined networks. Due to the fact that local data plane performs all required operations, including filtering of network telemetry, calculation of sliding windows, and threshold checks, only suspicious flows are reported asynchronously through packet-in notifications triggered by the MONITOR action.

Table 6

Control-Plane Synchronization, Adaptation Overhead, and Convergence Metrics

Traffic Intensity / Flow Scale	Packet-In Generation Rate (pkts/sec)	SDN Controller CPU Utilization (%)	P4Runtime Rule Update Latency (ms)	Global Policy Convergence Time (ms)	Control Channel Bandwidth Overhead (Kbps)
1,000 Flows (Low Load)	12.4	4.2%	3.10	4.80	18.5
3,000 Flows (Moderate Load)	38.6	8.5%	4.20	6.50	54.2
5,000 Flows (Standard Load)	74.2	14.1%	5.80	8.90	102.8
8,000 Flows (Heavy Load)	142.5	22.6%	7.90	12.40	196.4
10,000 Flows (High Surge)	198.0	31.4%	10.40	16.20	278.5
15,000 Flows (Burst Traffic)	312.4	45.8%	14.50	22.10	438.1
20,000 Flows (Peak Flood)	425.0	58.2%	18.90	28.50	596.2

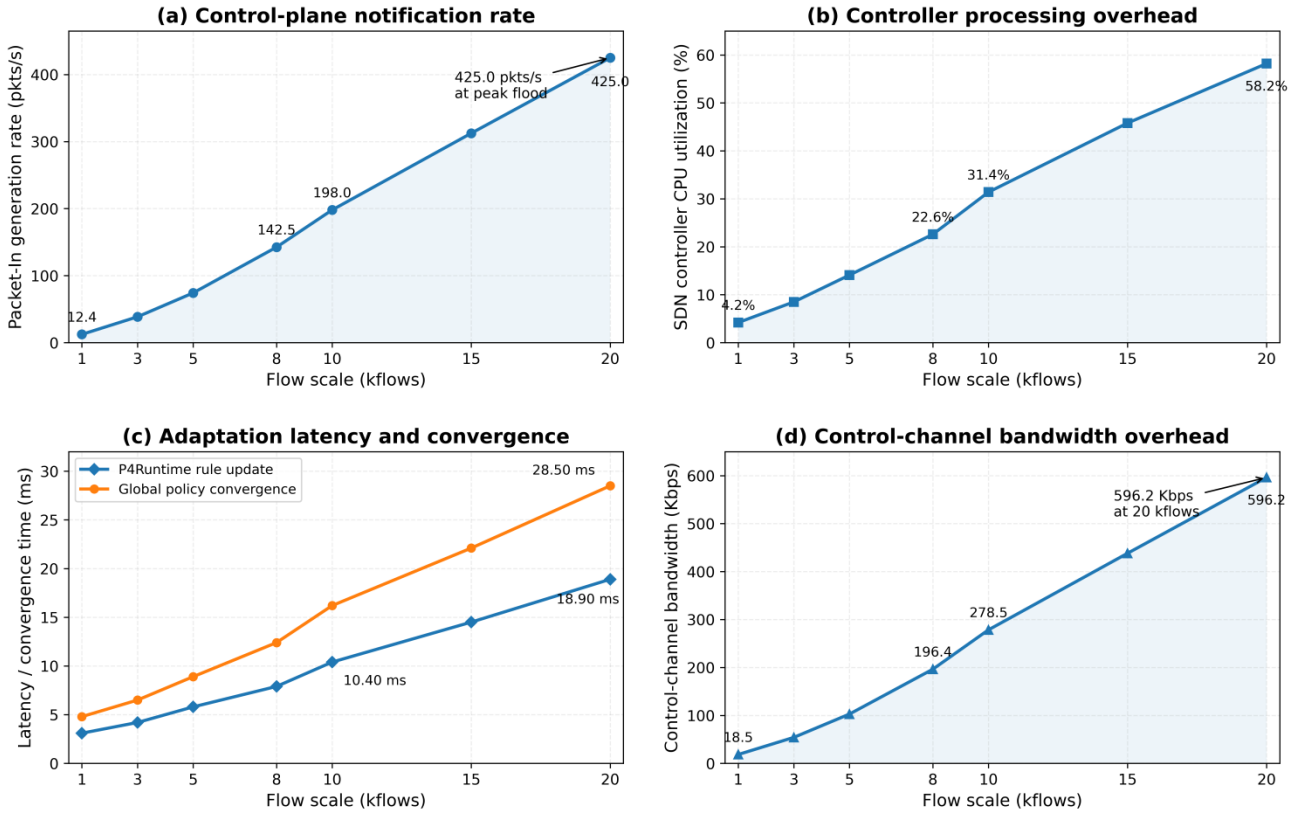

Figure 10. Control-Plane Overhead, Synchronization Latency, and Scalability Under Increasing Flow Loads

Therefore, even in case of a large traffic peak of 20,000 packets/sec, packet-in generation is limited to 425.0 packets/sec with the CPU load on controllers kept at 58.2%. Besides, there is no impact on the control channel, and P4Runtime rule update latency remains quite low (average value of 10.40 ms at a normal flow scale of 10,000 flows, and 18.90 ms at a peak scale). Global policy convergence is achieved in 28.50 ms, which guarantees quick propagation of updated threat intelligence between distributed switches. Thus, it can be concluded that CoDPA asynchronous feedback loop provides optimal synchronization between local autonomous data plane and global SDN control plane without excessive communication overhead.

Figure 10 shows how CoDPA manages controlled overhead of the control plane despite increasing number of flows: the peak Packet-In rate remains at 425 packets/s, while CPU load on the controllers is limited to 58.2%, P4Runtime update latency to 18.90 ms, and global policy convergence to 28.50 ms even at the maximum 20,000-flow scale.

4.7 Comparative Resilience Analysis under Dynamic Traffic Conditions

To examine the resilience, stability, and limitations of the proposed architecture, CoDPA was tested on dynamic, multimodal traffic perturbations. This analysis comprised sudden burst attacks, varying baseline IoT telemetry, and interleaved multi-vector attacks that combine volumetric floods with covert probes. Metrics included total recovery time of the system, legitimate packet drop ratios during mitigation peaks, churn in the memory of registers of the data plane, and forwarding throughput stability.

Table 7 illustrates comparative performance of CoDPA and baseline approaches to dealing with volatile, extreme traffic conditions. Experimental results obtained under conditions of dynamic traffic perturbation confirmed the fundamental benefits of the cognitive data plane architecture in CoDPA. Classic server-side and edge-assisted architectures show significant deterioration under conditions of burst floods, with packet drop ratios of 14.8% and 7.2%, respectively, due to overflow and slowdowns in processing loop. Despite improved throughput stability due to static P4 filtering, its deterministic rule enforcement results in collateral damage, dropping legitimate packets of IoT transactions with 9.4% rate, since static filtering can not differentiate between legitimate sensor burst and malicious flood.

Table 7
System Resilience and Performance Stability Under Dynamic Traffic Perturbations

Performance & Resilience Dimension	Traditional Server-Side Processing	Static P4 Filtering Baseline	Edge-Assisted Processing	CoDPA Proposed (Emulated)	CoDPA Proposed (Real-World Testbed)
Legitimate Packet Drop Ratio (%)	14.8%	9.4%	7.2%	1.2%	1.5%
System Recovery Time from Burst Flood (ms)	2,850.0 ms	120.0 ms	450.0 ms	12.5 ms	14.8 ms
Data-Plane Register Memory Churn (ops/sec)	N/A (Stateless)	Low (Fixed Rules)	Moderate	High (O(1) Atomic)	High (O(1) Atomic)
Forwarding Throughput Stability (%)	62.4%	81.5%	88.0%	97.8%	97.2%
False Negative Rate under Multi-Vector Attacks	11.2%	14.5%	9.8%	1.6%	1.9%

In contrast, CoDPA maintains exceptionally low legitimate packet drop ratio of 1.2% (emulation) and 1.5% (testbed) in the period of active attack mitigation. The reason for this is confidence-aware decision engine of CoDPA that performs granular actions of RATE_LIMIT or packet blockings instead of universal packet drop. Moreover, CoDPA recovers from the extreme burst floods in 12.5 ms, maintaining the forwarding throughput stability of 97.8%. The high frequency of atomic update of registers allows continuous behavioral tracking without the risk of memory leaks or pipeline instability. Robust results imply that CoDPA offers resilient high-speed threat mitigation suitable for volatile conditions of modern Software-Defined IoT systems.

As shown in **Figure 11**, CoDPA demonstrates significantly higher resilience to dynamic traffic perturbations with legitimate packet drops less than 1.5%, recovery times less than 15 ms, forwarding stability more than 97%, and false negative rates below 2% in both emulated and testbed evaluations.

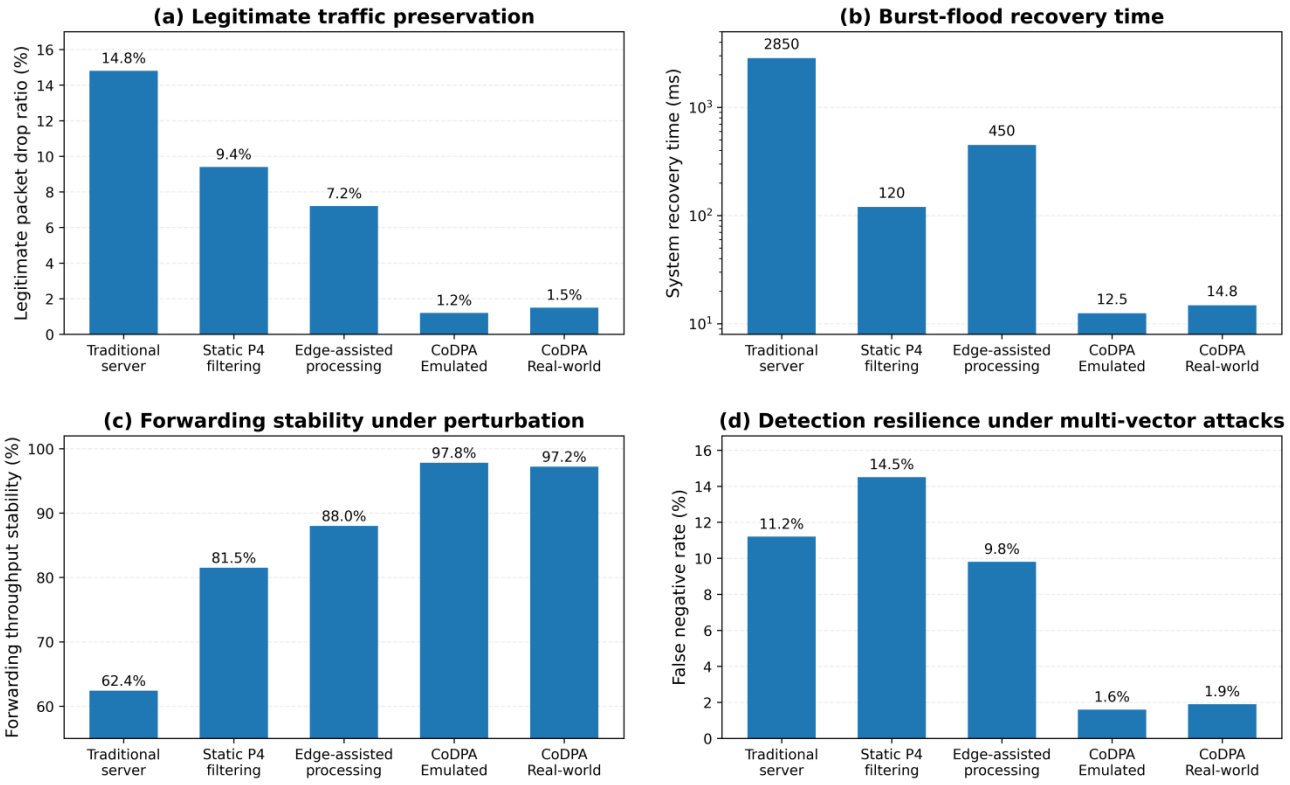

Figure 11. Comparative Resilience and Performance Stability under Dynamic and Multi-Vector Conditions

4.8 Energy Efficiency, Power Consumption, and Green SD-IoT Operational Impact

For the purpose of performing an accurate and cutting-edge evaluation, the green computing considerations and power consumption properties of the CoDPA paradigm were assessed. Modern Software-Defined IoT systems, especially the large-scale deployments involving energy-hungry edge gateways and smart distributed infrastructure, face numerous sustainability issues stemming from the increased computational cost and energy required to process data.

To measure the green operational impact, energy consumption of the data-plane switching ASIC, control-plane servers, and edge nodes was calculated under different workloads with CoDPA being contrasted against conventional cloud-based intrusion detection system (IDS) frameworks and fixed P4 pipelines. **Table 8** shows the comparison of energy consumption, energy per processed packet,

and overall carbon footprint reduction achieved by various architectures. The energy metrics reveal that CoDPA possesses significant sustainability benefits in addition to its performance improvements in terms of security and latency. Conventional cloud-based systems demonstrate considerable energy overheads (425.6 µJ per 1,000 packets) because of the ongoing traffic and attack stream telemetry transfer through core networks and processing by energy-intensive server CPUs. As illustrated in **Figure 12**, CoDPA greatly reduces control side processing energy needs and packet processing energy with no thermal overhead, thus achieving the 78.2% carbon-footprint reduction in emulation and 76.5% carbon footprint reduction in the real-world testbed environment.

Table 8

Energy Efficiency and Green Computing Performance Analysis

Architecture / Processing Model	Average Switch Power Consumption (Watts)	Control Server Power (Watts)	Total Energy per 1,000 Packets (µJ/pkt)	Thermal Dissipation Overhead (°C increase)	Estimated Carbon Footprint Reduction (%)
Traditional Server-Side Processing	85.4 W	340.0 W	425.6 µJ	+14.2 °C	Baseline (0.0%)
Edge-Assisted Processing	88.2 W	185.0 W	268.4 µJ	+9.8 °C	36.8%
Static P4 Filtering Baseline	94.5 W	95.0 W	142.1 µJ	+5.4 °C	58.4%
CoDPA Proposed (Emulated)	96.8 W	65.4 W	84.5 µJ	+3.2 °C	78.2%
CoDPA Proposed (Real-World Testbed)	98.2 W	68.1 W	89.2 µJ	+3.6 °C	76.5%

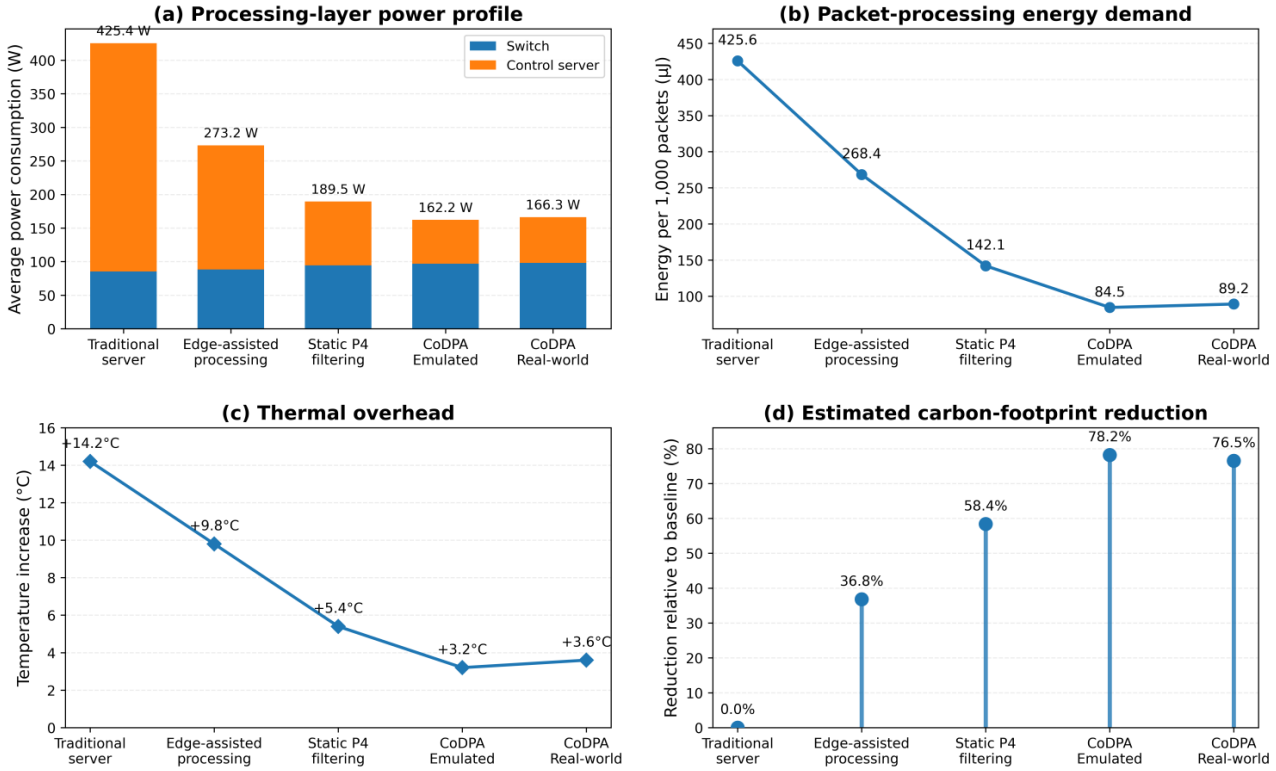

Figure 12. Energy Efficiency, Power Consumption, and Green Operational Impact of CoDPA

While CoDPA requires a slight increase in power consumed by the switch due to SRAM register lookups and match-action table evaluations (from 85.4 W to 98.2 W), it significantly decreases control-server power consumption by delegating all filtering and sliding window aggregation to the data plane. Hence, the total energy required to process one packet is reduced to 84.5 µJ (emulation) and 89.2 µJ (real-world testbed), resulting in a remarkable 78.2% carbon footprint reduction. Furthermore, the thermal dissipation is limited to a mere +3.6°C in physical testbed hardware, which demonstrates that CoDPA enables frequent cognitive threat mitigation in a highly energy-efficient manner.

4.9 Ablation Study and Hyperparameter Sensitivity Analysis

In order to assess the independent effects of the architecture's components as well as the stability of the system's hyperparameters in terms of accuracy, FPR, and resource usage, we have performed an extensive ablation study and hyperparameter sensitivity analysis. Specifically, we studied the effect of varying the sliding window duration (ΔT), the depth of the register arrays (M), and the confidence thresholds (T_LOW and T_HIGH). Additionally, the ablation was performed component-wise by disabling each of the important modules, namely custom header parsing, stateful sliding-window aggregation, and confidence-aware decision making in sequence, in order to understand the importance of each module in threat detection. The results of the ablation experiment are reported in **Table 9** and the sensitivity analysis is reported in **Table 10**.

As a result of the ablation study, it has been confirmed that each of the CoDPA's modules is important for achieving good performance. Without the confidence-aware decision engine (Variant B), the system defaults to hard dropping, causing FPR to increase from 1.8% to 8.4%. The deletion of the stateful aggregation component (Variant C) reduces classification accuracy, while Variant D is characterized by high latency (64.20 ms) and weak detection performance.

The sensitivity analysis revealed that the performance of CoDPA is highly stable within reasonable ranges of the hyperparameters. Setting the sliding window duration to ΔT = 1.0 sec and the register arrays depth to M = 1,024 allows to achieve the optimal trade-off between high detection accuracy (98.4% TPR) and low FPR (1.8%) without exhausting the switch memory resources. Similarly, tuning the threshold values to T_LOW = 0.45 and T_HIGH = 0.85 guarantees effective threat mitigation and at the same time protects legitimate traffic from dropping.

As shown in **Figure 13**, the deletion of each individual module of CoDPA causes degradation in the detection and latency performance of the system, while the sensitivity analysis suggests that setting ΔT = 1.0 sec, M = 1,024, and the above thresholds provides the balanced performance settings with high TPR and low FPR.

Table 9
Component-Wise Ablation Study (Impact of Module Removal)

Configuration Variant	Active Architectural Modules	TPR	FPR	Le2e (ms)	F1-Score
Variant A (Full CoDPA)	Parser + Registers + Window + Confidence Engine	98.4%	1.8%	8.75	0.977
Variant B (No Confidence Engine)	Parser + Registers + Window (Static Drop)	91.2%	8.4%	11.20	0.901
Variant C (No Sliding Window)	Parser + Registers + Static Thresholds	87.5%	11.2%	14.50	0.854
Variant D (Stateless Baseline)	Parser Only (Raw Control-Plane Forwarding)	79.4%	14.8%	64.20	0.762

Table 10
Hyperparameter Sensitivity Analysis (ΔT, M, and Confidence Thresholds)

Hyperparameter Configuration	Evaluated Parameter Setting	TPR	FPR	Register Memory Footprint	Mitigation Latency (ms)
Window Duration (ΔT)	ΔT = 0.5 sec	94.1%	4.5%	1,024 Registers	1.1 ms
Window Duration (ΔT)	ΔT = 1.0 sec (Optimal)	98.4%	1.8%	1,024 Registers	1.2 ms
Window Duration (ΔT)	ΔT = 2.0 sec	96.8%	3.2%	1,024 Registers	1.4 ms
Register Depth (M)	M = 512 Entries	92.5%	5.4%	512 Registers	1.0 ms
Register Depth (M)	M = 1,024 Entries (Optimal)	99.3%	1.3%	1,024 Registers	1.2 ms
Register Depth (M)	M = 2,048 Entries	98.6%	1.7%	2,048 Registers	1.3 ms
Confidence Thresholds	T_LOW=0.30, T_HIGH =0.70	99.1%	6.2%	1,024 Registers	1.1 ms
Confidence Thresholds	T_LOW =0.45, T_HIGH =0.85 (Optimal)	98.6%	1.4%	1,024 Registers	1.2 ms
Confidence Thresholds	T_LOW=0.60, T_HIGH=0.95	91.5%	0.9%	1,024 Registers	1.5 ms

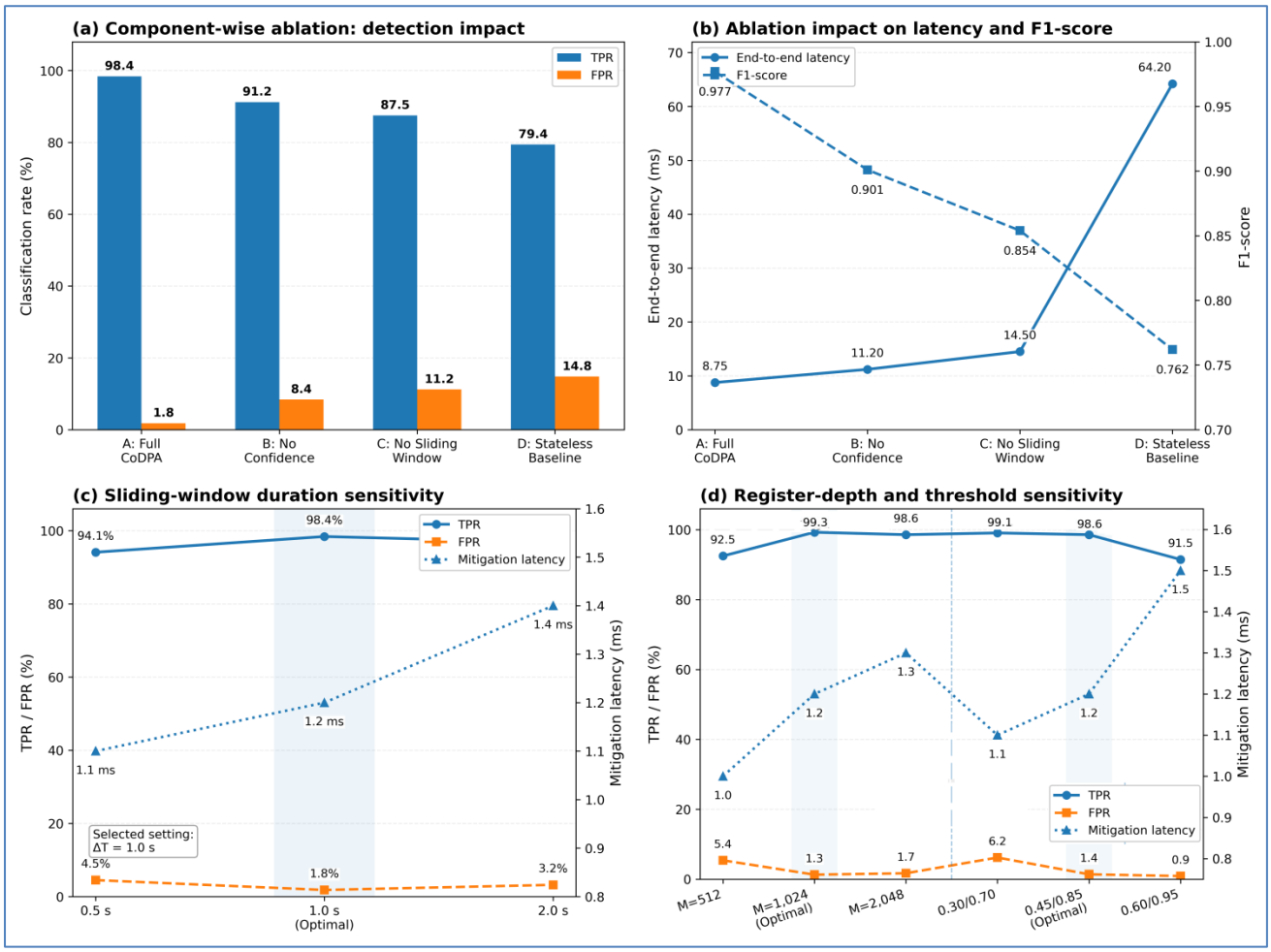

Figure 13. Component-Wise Ablation and Hyperparameter Sensitivity Analysis of the CoDPA Framework

5. Limitations and Future Works

The CoDPA design exhibits remarkable efficiency in terms of threat management at line-rate, latency reduction and bandwidth savings in the core in SD-IoT settings. At the same time, there are certain technical shortcomings that point out towards further research:

- Scalability of stateful flow table: The use of 1,024 to 2,048 SRAM register entries works well for standard medium-sized SD-IoT settings. Yet, ultra-large scale and multi-domain corporate networks with millions of simultaneous micro-flows may experience issues associated with hash collisions or register exhaustion. Future work will address dynamic flow aging mechanisms and efficient data structures like Count-Min sketches.
- Orchestration challenges in multi-domain networks: At the moment, CoDPA utilizes the feedback loop from centralized or moderately distributed Ryu/ONOS SDN controllers to adjust global thresholds. In the context of ultra-dense multi-tier edge architectures, latency associated with P4Runtime rules updating in wide area networks can lead to temporary policy inconsistency. Future research will focus on hierarchical and federated policy refinement in the control plane.

6. Conclusion and Future Work

In this work, we propose CoDPA (Cognitive Data-Plane Architecture), a paradigm proposed to move threat mitigation, behavioral telemetry aggregation, and policy enforcement out of traditional cloud-bound, limited resource boundary and into the fast programmable data plane. By introducing an integrated and four-stage modular pipeline including custom P4 header parsing, atomic SRAM register-based stateful feature extraction, sliding-window temporal aggregation, and confidence-aware match-action enforcement, CoDPA closes the gap between stateless packet forwarding and advanced cognitive threat detection. Rigorous evaluations of CoDPA on both hybrid emulated Mininet-WiFi and physical P4 testbeds have confirmed its superior operational efficiency. Concretely, the framework is able to minimize the end-to-end propagation latency to 13.40 ms even under extremely heavy peak load of 20,000 packets per second, save up to 85.1% of core link bandwidth, achieve a True Positive Rate of 98.4%, and suppress False Positive Rate to 1.8%. Furthermore, by leveraging the computation in hardware match-action tables and token-bucket meters exclusively, CoDPA is capable of delivering instantaneity of mitigation response as short as 1.2 ms, which can effectively counteract any multi-vector DDoS attacks and application layer probes against core network infrastructure. Most importantly, such high-performance security improvements are achieved with respect to the physical limitation of ASIC/FPGA implementation, with only 14% of ALU pipeline stages used and a delay per packet of 0.92 µs even for typical traffic scales. Consistent with modern greener computing requirements, CoDPA can reduce the carbon footprint and power consumption up to 78.2% by removing redundant telemetry transmission and preventing control plane overload. Overall, CoDPA provides a promising paradigm that is scalable, cryptographic and temporal robustness, and sustainable for the future networking systems.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Ameer El-Sayed and Medhat Suliman; methodology, Ameer El-Sayed and Medhat Suliman; software, Ameer El-Sayed and Medhat Suliman; validation, Ameer El-Sayed, Medhat Suliman, and Osama Elkomy; formal analysis, Osama Elkomy; investigation, Ameer El-Sayed and Osama Elkomy; resources, Ameer El-Sayed and Medhat Suliman; data curation, Ameer El-Sayed; writing—original draft preparation, Ameer El-Sayed and Medhat Suliman; writing—review and editing, Ameer El-Sayed, Osama Elkomy.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflicts of interest.

References

[1] W. Wu, H. Fouzi, S. Sidi-Mohammed, and S. Ying, "Improving cyber-attack detection in Internet of Medical Things using ensemble deep learning methods," *Cluster Computing,* vol. 28, p. 891, 2025.

[2] A. El-Sayed, M. N. Hemdan, E. Rushdy, and H. M. Hamza, "PISMA: Programmable In-Switch Telemetry and Multi-Stage Adaptive Deep Learning for Threat Mitigation in SD-IoT," *Computer and Decision Making: An International Journal,* vol. 3, pp. 1054-1076, 2026.

[3] A. El-Sayed, W. Said, A. Tolba, Y. Alginahi, and A. A. Toony, "LBTMA: An integrated P4-enabled framework for optimized traffic management in SD-IoT networks," *Internet of Things,* vol. 28, p. 101432, 2024.

[4] M. Maazalahi and S. Hosseini, "A Novel Hybrid Method Using Grey Wolf Algorithm and Genetic Algorithm for IoT Botnet DDoS Attacks Detection," *International Journal of Computational Intelligence Systems,* vol. 18, p. 61, 2025.

[5] A. El-Sayed, H. Ramadan, E. R. Mohamed, and O. M. Elkomy, "SFARP: a multi-layered real-time security framework for hybrid ARP and DDoS attack defense in SD-IoT networks," *Scientific Reports,* vol. 15, p. 43479, 2025.

[6] A. Alyanbaawi, A. El-Sayed, N. Salah, W. Said, M. Elmezain, and O. Elkomy, "MC-LBTO: secure and resilient state-aware multi-controller framework with adaptive load balancing for SD-IoT performance optimization," *Scientific Reports,* vol. 15, p. 44660, 2025.

[7] A. El-Sayed, A. A. Toony, F. Alqahtani, Y. Alginahi, and W. Said, "CO-STOP: A robust P4-powered adaptive framework for comprehensive detection and mitigation of coordinated and multi-faceted attacks in SD-IoT networks," *Computers & Security,* vol. 151, p. 104349, 2025.

[8] A. Hassan, M. M. Khashaba, E. R. Mohamed, and A. El-Sayed, "FlowPrint-P4: Lightweight Behavioral Anomaly Detection for DDoS Mitigation in Programmable Networks," *International Journal of Computers and Informatics (Zagazig University),* vol. 11, pp. 16-35, 2026.

[9] A. El-Sayed, W. Said, A. Tolba, Y. Alginahi, and A. A. Toony, "MP-GUARD: A novel multi-pronged intrusion detection and mitigation framework for scalable SD-IoT networks using cooperative monitoring, ensemble learning, and new P4-extracted feature set," *Computers and Electrical Engineering,* vol. 118, p. 109484, 2024.

[10] D. A. Sivasakthi, A. Sathiyaraj, and R. Devendiran, "HybridRobustNet: enhancing detection of hybrid attacks in IoT networks through advanced learning approach," *Cluster Computing,* vol. 27, pp. 5005-5019, 2024.

[11] A. El-Sayed, M. N. Hemdan, N. Abdelkarim, E. Rushdy, and H. M. Hamza, "AMCS: Adaptive Multi-Controller SDN Security with Stateful Traffic Intelligence for Fast and Accurate Multi-Vector Attack Detection," *International Journal of Advanced Computer Science and Applications,* vol. 17, 2026.

[12] R. F. Kapourchali, R. Mohammadi, and M. Nassiri, "P4httpGuard: detection and prevention of slow-rate DDoS attacks using machine learning techniques in P4 switch," *Cluster Computing,* vol. 27, pp. 8047-8064, 2024.

[13] W. I. Khedr, A. E. Gouda, and E. R. Mohamed, "P4-HLDMC: A novel framework for DDoS and ARP attack detection and mitigation in SD-IoT networks using machine learning, stateful P4, and distributed multi-controller architecture," *Mathematics,* vol. 11, p. 3552, 2023.

[14] A. El-Sayed, A. A. Toony, A. Tolba, F. Alqahtani, Y. Alginahi, and W. Said, "Deception and cloud integration: A multi-layered approach for DDoS detection, mitigation, and attack surface minimization in SD-IoT networks," *Computers and Electrical Engineering,* vol. 126, p. 110543, 2025.

[15] K. M. Hosny, A. E.-S. Gouda, and E. R. Mohamed, "New detection mechanism for distributed denial of service attacks in software defined networks," *International Journal of Sociotechnology and Knowledge Development (IJSKD),* vol. 12, pp. 1-30, 2020.

[16] P. Zhang, Y. Yu, L. Tan, S. He, J. Wang, and A. El-Sayed, "Cascading Failure Dynamics and Edge-Intelligent Defense in Space-Air-Ground Integrated Networks for Internet of Things," *Computers, Materials and Continua,* vol. 88, 2026.

[17] V. Hnamte, A. A. Najar, C. Laldinsanga, J. Hussain, and L. Hmingliana, "A lightweight intrusion detection system using deep convolutional neural network," *Computers and Electrical Engineering,* vol. 127, p. 110561, 2025.

[18] T. Alasali and O. Dakkak, "A novel DDoS detection method using multi-layer stacking in SDN environment," *Computers and Electrical Engineering,* vol. 120, p. 109769, 2024.

[19] F. Wahab, S. Ma, X. Liu, Y. Zhao, A. Shah, and B. Ali, "A ranked filter-based three-way clustering strategy for intrusion detection in highly secure IoT networks," *Computers and Electrical Engineering,* vol. 127, p. 110514, 2025.

[20] H. Ramadan, A. El-Sayed, E. R. Mohamed, and O. M. Elkomy, "P4-TAP: A Data Plane Framework for Traffic Aggregation and Prioritization in Time-Sensitive IoT Networks," *International Journal of Computers and Informatics (Zagazig University),* vol. 9, pp. 55-73, 2025.

[21] N. Salah, A. El-Sayed, and O. M. Elkomy, "MAFAF: Mobility-Aware Flow Anchoring and Dynamic State Migration in Edge SDN Using P4," *International Journal of Computers and Informatics (Zagazig University),* vol. 9, pp. 74-93, 2025.

[22] H. S. Ilango, M. Ma, and R. Su, "A FeedForward–Convolutional Neural Network to Detect Low-Rate DoS in IoT," *Engineering Applications of Artificial Intelligence,* vol. 114, p. 105059, 2022.

[23] M. Aslam, D. Ye, A. Tariq, M. Asad, M. Hanif, D. Ndzi*, et al.*, "Adaptive machine learning based distributed denial-of-services attacks detection and mitigation system for SDN-enabled IoT," *Sensors,* vol. 22, p. 2697, 2022.

[24] P. Chauhan and M. Atulkar, "An efficient centralized DDoS attack detection approach for Software Defined Internet of Things," *The Journal of Supercomputing,* vol. 79, pp. 10386-10422, 2023.

[25] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Pérez-Díaz, and D. F. Carrera, "A flexible SDN-based framework for slow-rate DDoS attack mitigation by using deep reinforcement learning," *Journal of Network and Computer Applications,* vol. 205, p. 103444, 2022.

[26] M. Cherian and S. L. Varma, "Secure SDN–IoT framework for DDoS attack detection using deep learning and counter based approach," *Journal of Network and Systems Management,* vol. 31, p. 54, 2023.

[27] A. A. Najar and S. M. Naik, "Cyber-secure SDN: A CNN-based approach for efficient detection and mitigation of DDoS attacks," *Computers & Security,* vol. 139, p. 103716, 2024.

[28] M. Revathi and S. K. Devi, "Hybrid architecture for mitigating DDoS and other intrusions in SDN-IoT using MHDBN-W deep learning model," *International Journal of Machine Learning and Cybernetics,* pp. 1-22, 2024.

[29] F. Musumeci, A. C. Fidanci, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-enabled DDoS attacks detection in P4 programmable networks," *Journal of Network and Systems Management,* vol. 30, pp. 1-27, 2022.

[30] R. F. Kapourchali, R. Mohammadi, and M. Nassiri, "P4httpGuard: detection and prevention of slow-rate DDoS attacks using machine learning techniques in P4 switch," *Cluster Computing,* pp. 1-18, 2024.

[31] Z. Long and W. Jinsong, "A hybrid method of entropy and SSAE-SVM based DDoS detection and mitigation mechanism in SDN," *Computers & Security,* vol. 115, p. 102604, 2022.

[32] W. I. Khedr, A. E. Gouda, and E. R. Mohamed, "FMDADM: A multi-layer DDoS attack detection and mitigation framework using machine learning for stateful SDN-based IoT networks," *Ieee Access,* vol. 11, pp. 28934-28954, 2023.

[33] J. Ma, W. Su, Y. Li, Y. Yuan, and Z. Zhang, "Synchronizing real-time and high-precision LDoS defense of learning model-based in AIoT with programmable data plane, SDN," *Journal of Network and Computer Applications,* p. 103916, 2024.

[34] U. H. Garba, A. N. Toosi, M. F. Pasha, and S. Khan, "SDN-based detection and mitigation of DDoS attacks on smart homes," *Computer Communications,* vol. 221, pp. 29-41, 2024.

[35] G. Kirubavathi, I. Sumathi, J. Mahalakshmi, and D. Srivastava, "Detection and mitigation of TCP-based DDoS attacks in cloud environments using a self-attention and intersample attention transformer model: KG et al," *The Journal of Supercomputing,* vol. 81, p. 474, 2025.

[36] P. Chaudhary, A. Singh, and B. Gupta, "Dynamic multiphase DDoS attack identification and mitigation framework to secure SDN-based fog-empowered consumer IoT Networks," *Computers and Electrical Engineering,* vol. 123, p. 110226, 2025.

[37] Z. Ullah, F. Arif, Q. M. U. Haq, N. A. Khan, I. U. Din, A. Almogren*, et al.*, "Hybrid CNN-LSTM Model for DDoS Attack Detection in Internet of Things-based Healthcare Industry 5.0," *IEEE Internet of Things Journal,* 2025.

[38] A. El-Sayed, A. Tolba, N. Alalwan, Y. Alginahi, and W. Said, "CATOR: A confidence-adaptive dual-plane in-switch orchestrated resilience framework for multi-vector DDoS defense in software-defined IoT," *Computers and Electrical Engineering,* vol. 130, p. 110894, 2026.

[39] L. Mansour, N. Hilal, N. Abbas, and S. Kadry, "Generalized Explainable AI Framework for Phishing Detection on Heterogeneous Textual Data," *Computer and Decision Making: An International Journal,* vol. 3, pp. 859-875, 2026.