AMCS: Adaptive Multi-Controller SDN Security with Stateful Traffic Intelligence for Fast and Accurate Multi-Vector Attack Detection

Ameer El-Sayed[1], Mohamed Nosseir Hemdan[2], Noha Abdelkarim[3], Ehab Rushdy[4], Hanaa M. Hamza[5]
Department of Information Technology-Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt[1, 2, 3, 4, 5]
Department of Information Technology-Faculty of Information Technology and Computer Science, Sinai University, Egypt[2]

Abstract*—The rapid proliferation of Software-Defined Networking (SDN) in large-scale Internet of Things (IoT) ecosystems has amplified exposure to sophisticated, multi-vector cyberattacks that simultaneously exploit control- and data-plane asymmetries. Existing single-controller statistical detectors, while effective under high-volume anomalies, fail to sustain precision and responsiveness under dynamic, distributed, or low-rate attack conditions. Addressing this critical gap, we propose AMCS—an Adaptive Multi-Controller SDN Security framework that fuses stateful traffic intelligence with cooperative inter-controller decision-making to enable resilient, context-aware detection across complex IoT traffic. AMCS embeds lightweight, P4-based stateful processing directly in the data plane and augments it with adaptive entropy-driven anomaly evaluation and consensus-based coordination among distributed controllers. Extensive experiments on an SDN–IoT testbed demonstrate that AMCS achieves up to 99.7% detection accuracy for high-volume floods, 96.8% for low-rate anomalies, and 94.1% under mixed traffic, while maintaining false-positive rates below 5% and detection latencies as low as 1.24 s. The cooperative consensus protocol enhances cross-controller reliability to 98.4% with only 0.83 s synchronization delay, while reducing control overhead by 34.7% compared to the single-controller baseline. Moreover, the distributed mitigation layer reacts within 1.6 s on average, neutralizing over 97% of attack flows with negligible collateral impact. Collectively, these results confirm that integrating stateful in-switch analytics, adaptive thresholding, and multi-controller cooperation establishes a scalable, self-adaptive SDN security fabric—achieving both fast detection and stable defense against evolving multi-vector threats in IoT-driven networks.

Keywords*—SDN; IoT; multi-controller security; stateful traffic analysis; adaptive anomaly detection; distributed mitigation; entropy-based indicators; cooperative defense; P4 programmable networks; cyber-physical systems security

I. Introduction

The rapid expansion of the Internet of Things (IoT) has reshaped modern network infrastructures, introducing unprecedented levels of heterogeneity, mobility, and traffic dynamism [1]. Although Software-Defined Networking (SDN) offers a programmable and logically centralized architecture capable of managing this complexity, its increasing adoption in IoT and cloud-integrated settings has exposed several structural vulnerabilities [2]. In particular, current SDN deployments depend heavily on the interaction between the control and data planes, making the controller a critical point whose performance can be disrupted by malicious traffic surges or crafted low-rate attacks [3]. These concerns are well documented in prior studies, which highlight how Distributed Denial-of-Service (DDoS) campaigns, spoofed traffic, and saturation attacks degrade controller responsiveness and accuracy, often overwhelming existing statistical detection methods that rely on basic entropy or rate measurements [4].

Recent research further emphasizes that the challenge extends beyond traditional high-volume attacks [5]. In IoT-dense environments, the number of devices and the irregularity of their traffic patterns increase the difficulty of distinguishing benign variations from genuine anomalies [6]. This complexity is compounded in multi-controller SDN architectures—designed to improve scalability and fault tolerance—where controllers must coordinate decisions, synchronize state, and maintain consistent security policies across distributed domains [7]. Without robust mechanisms for inter-controller cooperation, these architectures may inadvertently widen the attack surface or delay collective responses to emerging threats [8]. There is a growing need for secure, coordinated, and intelligent controller communication in SDN networks, which outlines challenges such as secure coordination, attack-surface expansion, and the limitations of stateless detection pipelines in fast-evolving IoT scenarios [9].

While previous work demonstrates that entropy-based methods can achieve high detection accuracy in controlled conditions for specific attack models, these techniques often operate independently on each controller and lack a mechanism to incorporate deeper traffic state or contextual behavior [10], [11]. As a result, statistical fluctuations, low-rate probing, and distributed multi-vector attacks may evade or delay detection [12]. The limitations documented in earlier SDN studies point to a broader gap: existing single-controller, threshold-driven systems are not designed to reason about traffic evolution over time, nor to leverage cross-controller observations that could reveal subtle, otherwise-hidden attack patterns [13].

Motivated by these challenges, this work explores an adaptive multi-controller security framework that integrates stateful traffic inspection with cooperative intelligence sharing. The proposed approach integrates multi-layer protection, real-time stateful monitoring, secure controller-to-controller communication, and seamless orchestration of distributed defensive functions across the network. By combining stateful flow context with coordinated multi-controller analysis, the

proposed framework aims to enhance detection robustness against diverse attack strategies—including low-rate exfiltration attempts, distributed probing, and rapid DDoS-like bursts—while maintaining the performance and scalability demands inherent to IoT-driven networks.

II. Literature Review

The convergence of SDN and the IoT has reshaped modern network management, offering programmability, centralized orchestration, and policy-driven control. However, this integration also exposes new vulnerabilities. The centralized nature of SDN controllers and the heterogeneous, resource-limited behavior of IoT nodes collectively create an expanded attack surface, particularly against multi-vector cyberattacks. Maintaining the stability of SD-IoT environments under such conditions has therefore become a cornerstone of current research, as network-layer resilience directly underpins higher-level security assurances.

Early contributions in this domain largely focused on controller-centric machine learning (ML) approaches for attack detection. Adaptive ensemble frameworks and hybrid classifiers achieved notable accuracy in simulated testbeds but frequently failed to generalize beyond controlled datasets [14]. For instance, neural-network–SVM hybrids demonstrated high detection performance on benchmark datasets but suffered from limited scalability and dataset bias [15]. Other lightweight gradient-boosting models improved training efficiency yet still relied heavily on handcrafted features and synthetic traffic patterns, reducing their practicality for real IoT deployments [16].

Subsequent research explored deep learning (DL) and reinforcement learning (RL) paradigms to enhance automation and adaptiveness [17]. Architectures that combined DL-based detectors with RL-driven mitigation improved dynamic decision-making but were constrained by computational overheads and untested scalability in multi-controller topologies [18]. CNN- and DBN-based frameworks [19], [20] produced high detection rates; however, their limited interpretability and dependence on non-IoT datasets restricted their utility in heterogeneous traffic environments.

Parallel to controller-side innovation, attention shifted toward programmable data-plane intelligence. The advent of P4 switches enabled in-network detection, where selected traffic features are processed closer to the source. P4-assisted methods demonstrated substantial latency reduction and early anomaly recognition, particularly for TCP- and HTTP-based floods [21], [22]. Nevertheless, computational constraints within the data plane and narrow attack coverage remained unresolved. Meanwhile, controller-integrated ML frameworks—such as SDN-WISE [23] and DALCNN [24]—offered improved localized detection accuracy but continued to suffer from single-point bottlenecks and dependence on static detection thresholds.

A later wave of research introduced hybrid statistical–ML approaches to balance efficiency and adaptability. Models combining entropy-based indicators with machine learning improved early detection of low-rate anomalies. Similarly, edge-level designs leveraging Markov models and hybrid deep networks achieved effective performance in mitigating slow-rate IoT botnet traffic [25]. Yet, these methods often lacked flexibility under variable attack rates, emphasizing the need for dynamic temporal analysis and context-aware feature extraction [26], [27]. The shift toward adaptive temporal windows and confidence-based learning mechanisms represents a direct response to these limitations [28].

More recent studies have extended this evolution into context-aware, multi-phase, and transformer-based architectures. Attention-driven models such as SAINT [29] and multiphase entropy–clustering frameworks demonstrated high detection accuracy (above 96%) in cloud and fog environments [30]. Comparative evaluations [31] reaffirmed the efficiency of traditional ML classifiers—like Decision Trees and Random Forests—as practical baselines for constrained devices, while hybrid interpretable approaches integrated explainability tools such as SHAP for domain-specific applications (e.g., healthcare IoT) [32]. Nonetheless, the majority of existing work remains tied to isolated control domains or non-IoT datasets, limiting generalization across distributed and dynamic infrastructures [33].

Despite notable advances, few existing frameworks achieve real dual-plane coordination—that is, combining stateful in-switch analytics with adaptive learning and distributed mitigation across multiple controllers. As large-scale DDoS attacks remain the most frequent and destructive threat to IoT ecosystems, building resilient, cooperative, and self-adaptive SD-IoT defenses remains an open and urgent research frontier.

The proposed framework in this study contributes to this effort by integrating stateful data-plane inspection, entropy-aware adaptive detection, and multi-controller collaborative mitigation. Table I summarizes the representative literature, highlighting trends, strengths, and research gaps that informed the design of the proposed solution.

TABLE I. Summary of Representative Research on Attack Detection and Mitigation in SD-IoT Networks

Ref.	Year	Approach	Main Strengths	Identified Limitations
[17]	2022	SDN framework with DL detection + RL mitigation	Dynamic, adaptive countermeasures	Scalability not validated in real SD-IoT settings
[15]	2022	FFCNN–SVM hybrid for low-rate DDoS	Accurate under controlled conditions	Single attack focus, limited dataset diversity
[24]	2022	DALCNN integrated with SDN controller	High accuracy and stability	Dependent on non-IoT traffic datasets
[21]	2022	P4-enabled ML-based detection	Low latency, edge-level monitoring	Narrow attack scope, heavy switch load
[27]	2022	Entropy + ML hybrid model	Fast and precise detection	Limited handling of multi-vector attacks

[18]	2023	DL-driven SDN defense with adaptive responses	Integrative multi-stage mitigation	Lacks IoT-specific validation
[28]	2023	FMDADM using packet-window features	Effective IoT botnet detection	Limited adaptability to variable traffic rates
[23]	2023	SDN-WISE ML-based detection	Improved localized accuracy	Single-controller dependency
[16]	2023	LightGBM-based feature engineering	Efficient and lightweight	Synthetic data reliance
[25]	2024	Edge-level hybrid deep learning	Robust low-rate attack detection	Scalability is unproven in large topologies
[26]	2024	Smart-home ML + signature detection	High accuracy for specific IoT domains	Poor generalization to other IoT environments
[22]	2024	P4-based HTTP DDoS defense	Low latency, real-time operation	High computational cost on P4 devices
[29]	2025	Transformer (SAINT) model for TCP-DDoS	Highly interpretable, scalable	Focused mainly on transport-layer floods
[30]	2025	Multiphase ML–entropy fog architecture	High classification accuracy, quick response	Evaluation restricted to fog nodes
[34]	2025	CNN–LSTM with SHAP interpretability	Transparent DL behavior	Limited to the healthcare IoT context
[33]	2025	ONOS Flood Defender with adaptive algorithms	Real-time SYN flood mitigation	Fixed statistical thresholds, limited adaptability
This Work	**2025**	P4-based SD-IoT framework with adaptive detection and multi-controller mitigation	Real-time stateful feature extraction, confidence-adaptive learning, scalable coordination	Requires further testing under large-scale, distributed failover conditions

III. Proposed Framework

This section presents the proposed adaptive security framework for multi-controller SDN environments supporting heterogeneous IoT deployments. The framework addresses core challenges in scalability, control-plane coordination, and adaptive security monitoring by combining stateful traffic analysis, cooperative controller synchronization, and distributed mitigation. It is structured around four main components:

1) The system model, describing the architectural layers and interactions between network entities;

2) The threat model, identifying relevant attack vectors and their potential impact;

3) The stateful traffic-processing and adaptive anomaly evaluation modules, enabling context-aware detection; and

4) The multi-controller coordination and response mechanisms, which provide resilience and consistent defense actions.

Section 3.1 introduces the overall system architecture, while Section 3.2 defines the threat landscape and attacker capabilities. The following subsections detail the analytical and cooperative modules that enable detection, decision-making, and mitigation within the distributed control environment.

A. System Model

The proposed Adaptive Multi-Controller SDN Security (AMCS) framework (Fig. 1) embodies a hierarchical yet cooperative architecture that unifies stateful data-plane intelligence with distributed controller coordination. It is composed of three principal tiers and eight logically connected functional modules that collectively enable programmable, adaptive, and synchronized protection across large-scale IoT networks.

(1) IoT and Edge Layer

At the edge of the architecture lies a diverse ecosystem of IoT devices—sensors, cameras, gateways, and embedded controllers—that generate heterogeneous traffic streams. These devices (Step ① in Fig. 1) transmit telemetry, command, and control data toward the SDN infrastructure. Their irregular bursty transmissions and constrained resources make them vulnerable entry points for both volumetric and stealthy threats such as spoofing, infiltration, and low-rate exfiltration.

(2–5) Data-Plane Intelligence: Stateful Feature Extraction and Analysis

Incoming traffic first enters the SDN data plane (Step ②), composed of P4-programmable switches capable of executing local inspection before involving the controller. Each switch captures and extracts essential flow-level attributes—packet size, inter-arrival time, flow initiation rate, and destination entropy (Steps ③–④)—which are forwarded to the Stateful Traffic-Processing Module (STPM) located inside the P4 pipeline (Step ⑤).

The STPM maintains per-flow state tables and statistical registers that continuously update as packets traverse the switch. This enables in-network anomaly sensing by identifying temporal irregularities in packet behavior, thereby offloading early detection from the control plane. By exporting only aggregated summaries rather than raw packets, the STPM significantly reduces packet in congestion and controller CPU load.

(6) Control-Plane Intelligence: Adaptive Anomaly Evaluation (AAE)

Processed telemetry from STPM is sent to the controller's Adaptive Anomaly Evaluation (AAE) module (Step ⑥). The AAE dynamically computes entropy-based deviation scores and adjusts its thresholds using real-time variance and historical profiles. This allows the controller to distinguish flash-crowd events from true anomalies. Each controller thus maintains a localized, context-aware anomaly model that evolves with observed traffic behavior.

(7) Inter-Controller Coordination: Multi-Controller Coordination Module (MCCM)

Because the AMCS framework deploys multiple distributed controllers across domains (C1, C2 … Cn), synchronization of

local detections is essential. The MCCM (central-CO exchange shown in Fig. 1, Step ⑦) aggregates alerts from all controllers, performs consensus voting based on trust weights, and disseminates unified decisions across peers. Consensus cycles occur periodically, ensuring global consistency without imposing excessive communication overhead. This cooperative verification minimizes false positives and enables rapid multi-domain agreement—typically within 0.8 s, as shown in experimental results.

(8) Global Mitigation and Policy Enforcement: Distributed Mitigation Workflow (DMW)

Once the MCCM confirms an incident, the Distributed Mitigation Workflow (DMW) (Step ⑧) executes adaptive countermeasures proportionate to alert confidence and severity. Controllers enforce localized rate-limiting, flow blocking, or rerouting policies within their domains, while global synchronization ensures that legitimate flows remain unaffected. This distributed enforcement loop closes the detection–response cycle, maintaining network stability and service continuity.

The bidirectional arrows between modules denote feedback loops: mitigation outcomes refine AAE thresholding, and updated trust weights from MCCM influence future coordination cycles. Together, these interactions form a closed, adaptive control system that continually tunes its detection sensitivity, synchronization accuracy, and policy strength.

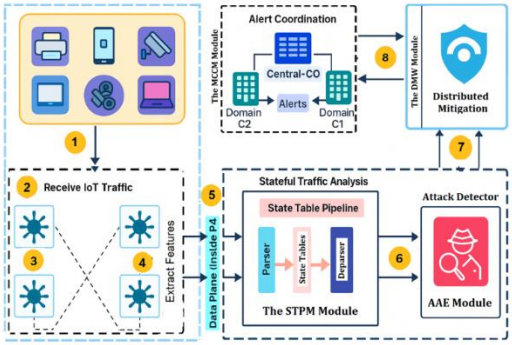

Fig. 1. System model of the AMCS framework showing end-to-end data flow from traffic ingestion to distributed mitigation.

B. Threat Model

The threat model defines the adversarial assumptions, potential attack vectors, and corresponding defense mapping relevant to the proposed AMCS framework. It considers an intelligent, adaptive adversary capable of strategic coordination across IoT domains, yet without direct access to the SDN controller software or administrative privileges. The attacker's objectives are to disrupt network availability, degrade controller responsiveness, and exploit synchronization weaknesses in multi-controller SDN environments.

As illustrated in Fig. 2, the threat model encompasses three logical dimensions: (1) Adversary Capabilities, (2) Attack Vectors and Impacted Assets, and (3) Defense Mapping to the corresponding AMCS protection modules.

To counter these coordinated threats, the AMCS architecture implements a layered defense mapping between identified attack vectors and specific protection modules (as shown in Table II). Table III summarizes the threat characteristics and system impacts under the assumed adversarial model. The mapping clearly demonstrates that AMCS establishes a defense continuum—from in-switch detection (STPM) to global mitigation (DMW)—capable of addressing both volumetric and adaptive multi-vector attacks.

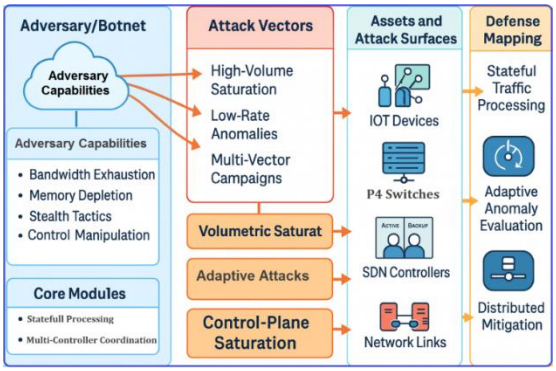

Fig. 2. Threat model showing adversarial capabilities, targeted SDN–IoT assets, corresponding attack vectors, and mapped AMCS defense mechanisms.

TABLE II. DEFENSE MAPPING TO AMCS MODULES

Threat Category	Affected Asset(s)	Primary Defense Mechanism
High-Volume Saturation	P4 Switches, Controllers	Stateful Traffic Processing (STPM) detects abnormal packet bursts directly within the data plane, throttling early-stage floods before escalation.
Low-Rate Stealth Attacks	IoT Devices, Network Links	Adaptive Anomaly Evaluation (AAE) dynamically adjusts detection thresholds using entropy variance and temporal context to capture slow anomalies.
Multi-Vector Distributed Campaigns	Multiple Controllers, Domains	Multi-Controller Coordination Module (MCCM) correlates alerts and ensures synchronized detection decisions across domains.
Control-Plane Manipulation	Controllers, State Tables	Distributed Mitigation Workflow (DMW) enforces coordinated policy adaptation, restoring consistent states across controllers and minimizing response delay.

TABLE III. THREAT MODEL SPECIFICATIONS

Threat Category	Adversary Capability	Impact on System
High-Volume Saturation	Flooding traffic to overload switches/controllers	Excessive *packet_in* load, delayed flow setup, and reduced throughput
Low-Rate Stealth Attacks	Gradual, subtle anomalies	Evades fixed thresholds, increases false negatives
Multi-Vector Distributed Attacks	Coordinated anomalies across domains	Inconsistent detection, delayed mitigation
Control-Plane Manipulation	Triggering conflicting flow behaviors	State inconsistency, incorrect routing, degraded QoS

C. Stateful Traffic-Processing Module (STPM)

Headings: The stateful traffic-processing module forms the analytical core of the framework. Its role is to extract concise indicators from live traffic, maintain short-term and long-term behavioral states, and provide each controller with context-rich signals that improve anomaly detection. Traditional SDN switches operate in a stateless manner, forwarding packets based solely on flow rules. This stateless behavior limits their ability to recognize deviations in flow evolution, protocol usage, or cross-packet characteristics. To overcome these limitations, the proposed module leverages programmable data-plane capabilities—implemented using P4—to embed lightweight stateful operations directly into the forwarding pipeline.

The processing logic follows a structured pipeline, as indicated in Fig. 3. Incoming packets are first parsed to extract protocol headers and application-layer fields relevant to DNS, HTTP, or other IoT-centric protocols. The parser feeds these fields into a match–action stage where the switch applies lookup rules while also performing state updates using registers and counters. These structures enable the switch to track properties such as packet size deviations, flow creation rates, destination diversity, and inter-arrival timing patterns. By maintaining state across packets, the switch can signal abnormal trends long before they result in high-volume anomalies or controller overload.

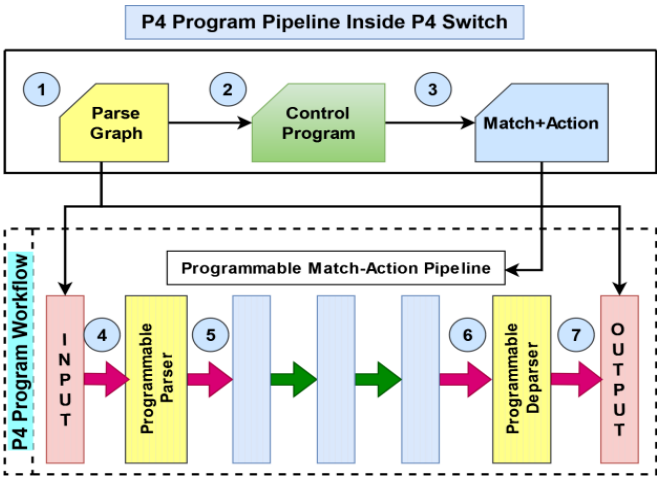

Fig. 3. Internal architecture and P4 pipeline implementation for stateful traffic-processing module (STPM).

A key advantage of this approach is that traffic summarization occurs in-network, reducing the controller's analytical burden. Instead of transmitting every suspicious packet, the switch periodically exports compact telemetry reports, such as deviations in feature values, sudden expansion of destination entropy, or sustained increments in per-flow byte counters. The module, therefore, blends immediate packet-level inspection with incremental state accumulation, producing a richer and more stable detection signal than stateless indicators.

The features selected for extraction reflect traffic characteristics known to fluctuate significantly under both high-volume and low-rate attack conditions. These features capture both structural packet properties and temporal flow behavior. Table IV summarizes the main features processed by the module and the corresponding P4 elements used to extract and maintain their state.

TABLE IV. EXTRACTED STATEFUL FEATURES AND P4 PIPELINE COMPONENTS

Feature	Description	P4 Element Used	Statefulness
Packet Size Deviation	Tracks changes in packet payload sizes over time, useful for spotting tunneling or abnormal payload inflation	register, counter	Maintains running average and variance
Flow Initiation Rate	Measures the frequency of new flow creation to detect sudden expansion typical of scans or DDoS bursts	register, metadata, digest	Stores timestamps and increments counters
Destination IP Spread	Tracks the diversity expansion of outbound destinations to reveal distributed probing	register array indexed by destination prefix	Persistent across time windows
Inter-Arrival Time	Measures packet timing irregularities indicative of slow-rate or stealth attacks	timestamp, register	Stores last-seen time per flow
DNS Query Length	Flags abnormal query sizes often associated with exfiltration tunnels	Custom P4 parser fields	Stateful threshold comparison
Byte Count per Flow	Tracks cumulative bytes to detect gradual data leakage	counter per flow	Persistent accumulation
Protocol Distribution	Monitors shifts in protocol usage, such as sudden surges in UDP flows	Table-level statistics, counter	Window-based aggregation
Source Entropy Indicators	Measures variability in the source distribution used during volumetric anomalies	Aggregated via digest to controller	Semi-stateful (derived from exports)

Stateful processing significantly enhances the switch's analytical capability. Rather than relying on instantaneous metrics, the module captures temporal evolution, which is critical for identifying low-rate deviations and subtle behavioral drift. Attacks that unfold gradually—such as data exfiltration, incremental scanning, or distributed low-volume campaigns—often evade purely statistical detectors. Maintaining state allows the system to detect cumulative anomalies, such as persistent shifts in packet size distributions, slow increases in destination diversity, or sudden escalations in flow initiation.

The design ensures that the state is both local and lightweight. Registers store compact numeric values indexed by flow or prefix, enabling O (1) updates while preserving forwarding speed. Counters accumulate features with minimal overhead. Furthermore, because periodic summaries rather than raw packets are exported, the control plane avoids congestion even in dense IoT environments.

Across the detection pipeline, stateful analysis provides three distinct advantages. First, it increases robustness by

smoothing short-term statistical noise. Second, it enhances sensitivity to low-rate or stealthy patterns. Third, it allows each controller to reason about local behavior using richer context before participating in cooperative, multi-controller verification. This combination of local state and global coordination strengthens the overall resilience of the network against diverse and evolving attack strategies.

As indicated in Algorithm 1, the Stateful Traffic-Processing Module operates directly within the data plane and incrementally maintains behavioral context for each flow. Its three-phase structure ensures that initialization sets stable analytical parameters, packet-level operations apply lightweight stateful logic, and telemetry summaries are efficiently exported without overwhelming the controller.

By combining inter-arrival tracking, byte accumulation, destination-spread evolution, and dynamic packet-size statistics, the module produces a refined view of traffic behavior long before anomalies manifest at larger scales. This approach reduces controller overhead and enables early identification of threats such as low-rate scans, slow-drip exfiltration, or sudden distribution shifts in outbound destinations. The final summary phase ensures that rich but compact flow-level insights are delivered upstream for coordinated decision-making.

```
Algorithm 1: Stateful Traffic-Processing Module (STPM)
Inputs:
      - IncomingPackets
      - FeatureThresholds
      - TelemetryWindow
      - SwitchStateTables
      - TimeStamps

Outputs:
      - Flow-level state summaries
      - Local anomaly flags
      - Compressed telemetry reports

BEGIN
    WHILE switch is active DO
        FOR each packet p ∈ IncomingPackets DO
            header ← ParsePacket(p)
            features ← ExtractFeatures(header)
            flowID ← IdentifyFlow(p)
            state ← LoadState(flowID)

            Stateful Updates
            state.byteCount ← state.byteCount + features.size
            state.IAT ← features.timestamp – state.lastSeen
            state.sizeStats ← UpdateSizeStatistics
            state.dstSpread ← UpdateSpreadMap(features.dstIP)

            Local Anomaly Check
            anomaly ← EvaluateLocalAnomaly(FeatureThresholds)
            IF anomaly == TRUE THEN
                GenerateLocalAlert(flowID, state)
                StoreState(flowID, state)
            END IF
        END FOR

        Telemetry Window Completion
        IF TelemetryWindowExpired() THEN
            report ← BuildTelemetry(SwitchStateTables)
            ExportTelemetry(report)
            ResetTelemetryWindow()
        END IF
    END WHILE
END
```

D. Adaptive Anomaly Evaluation Module

Accurate detection in dynamic SDN–IoT environments requires more than static comparison of feature values against fixed thresholds. Traditional statistical detectors often misclassify benign surges or clustered traffic as anomalies, and may also miss low-rate attacks that evolve gradually. These limitations have been well-documented, particularly the difficulty of distinguishing flash crowds from genuine attack traffic or differentiating short legitimate flows from spoofed ones. Moreover, entropy or feature-based indicators can fluctuate under heterogeneous traffic conditions, leading to false positives or delayed detection unless thresholds adapt to the observed traffic behavior. Several studies highlight that adaptive thresholding, especially when combined with multi-feature signals, improves stability and avoids misdiagnosis during both low-rate and distributed attack scenarios.

```
Algorithm 2: Adaptive Anomaly Evaluation (AAE)
Inputs:
      - FlowSummaries
      - LocalAlerts
      - HistoricProfiles
      - EntropyWindow
      - AdaptiveThresholdParameters

Outputs:
      - Updated anomaly threshold
      - Confirmed anomaly events
      - Controller-level alert flags

BEGIN
    # Phase 1 – Telemetry Consolidation
    mergedSummaries ← Merge(FlowSummaries, LocalAlerts)
    currEntropy ← ComputeEntropy(mergedSummaries, Entr_Window)
    baselineEntropy ← HistoricProfiles.recentEntropy
    entropyVariance ← Variance(HistoricProfiles.entropyHistory)
    drift ← |currEntropy – baselineEntropy|

    # Phase 2 – Adaptive Threshold Adjustment
    IF drift > entropyVariance THEN
        threshold ← AdjustThresholdUp(AdaptiveThre_Params)
    ELSE
        threshold ← AdjustThresholdDown(AdaptiveThre_Params)
    END IF
    UpdateThreshold(threshold)

    # Phase 3 – Anomaly Confirmation
    FOR each flow f in mergedSummaries DO
        score ← WeightedScore(
                    f.entropyDrop,
                    f.IAT,
                    f.dstConcentration
                )
        IF score ≥ threshold THEN
            ConfirmAnomaly(f)
            ShareAlert(f)
        ELSE
            ContinueNormal(f)
        END IF
    END FOR
END
```

The module operates in three stages. First, it consolidates telemetry from all switches, forming a global snapshot of flow-level behavior. Second, it evaluates anomalies using a weighted scoring model that accounts for entropy drop rate, inter-arrival irregularity, flow initiation bursts, and destination concentration ratios—metrics demonstrated to be reliable

under the entropy-based detection studies recorded in the previous work. Third, it dynamically adjusts the threshold based on the stability of these metrics. When legitimate and attack traffic characteristics converge—as demonstrated in multiple traffic models B and C—the adaptive mechanism stabilizes the detection rate by adjusting the decision boundary to reflect the mixed nature of traffic. To support this behavior, the module follows the operational logic summarized in Algorithm 2.

The AAE algorithm (Algorithm 2) refines the raw indicators produced by STPM into more robust and reliable anomaly decisions. Its input includes aggregated entropy trends, destination-spread divergence, and inter-arrival time irregularities. These signals align with the documented behaviors of both normal and attack traffic in SDN–IoT environments: for example, sharp drops in entropy correlate strongly with attack bursts, while mild fluctuations may represent flash crowds or active endpoints.

In Phase 1, the controller consolidates telemetry across multiple switches, forming a unified analytical surface. This ensures the evaluation process recognizes distributed effects that would be invisible to per-switch detection alone. In Phase 2, the adaptive thresholding mechanism adjusts the detection boundary by analyzing the direction and magnitude of entropy drift relative to its historical variance. This method directly addresses conditions observed in the experiment results, where similar traffic patterns produced different entropy values across traffic models A, B, and C, and where adaptive thresholding sustained detection rates above 93% even under complex flows.

Finally, Phase 3 applies weighted scoring to detect anomalies. By integrating multiple features rather than relying solely on entropy, the evaluation process becomes less sensitive to noise and better aligned with multi-vector attack characteristics. Confirmed anomalies are shared across controllers to enable synchronized response in multi-controller environments.

E. Multi-Controller Coordination Module (MCCM)

Accurate Large-scale SDN deployments increasingly adopt distributed control architectures to improve scalability, resilience, and latency. While multiple controllers enhance responsiveness, they introduce a crucial coordination challenge: ensuring that each controller maintains a consistent and synchronized view of network security state. Without proper coordination, local detections may remain isolated, delaying collective mitigation or generating conflicting control actions. The Multi-Controller Coordination Module (MCCM) resolves these inconsistencies by enabling real-time collaboration, state sharing, and consensus-driven anomaly verification among distributed controllers.

The MCCM acts as a synchronization layer above individual anomaly evaluators. Each controller operates independently on its local traffic domain using the Adaptive Anomaly Evaluation (AAE) process, but periodically exchanges summarized alerts and trust-weighted anomaly scores with peer controllers. This exchange allows for correlation of distributed evidence—such as simultaneous entropy drops, synchronized flow expansions, or shared destination clusters—indicating a coordinated or multi-vector attack. Through a combination of message aggregation, voting, and dynamic trust adjustment, controllers reach a joint decision on whether to escalate an alert into a confirmed global incident.

Coordination operates over lightweight communication channels, often gRPC or message-bus based, ensuring minimal delay. Each controller maintains a peer state table containing confidence weights and last-synchronization timestamps for all neighboring controllers. During collaboration, alerts are cross-validated using these weights, allowing the system to emphasize highly reliable peers while minimizing false confirmations from unstable nodes. Algorithm 3 presents the operational workflow of this module.

Algorithm 3: Multi-Controller Coordination Module (MCCM)

```
Inputs:
      - LocalAlerts
      - PeerControllersList
      - TrustWeights
      - SyncInterval
      - GlobalAlertThreshold

Outputs:
      - Verified global anomaly events
      - Updated trust table
      - Consensus alerts to mitigation layer

BEGIN
     # Phase 1 – Synchronization Cycle
     WHILE controller_is_active DO
          IF SyncIntervalExpired() THEN
               FOR each peer ∈ PeerControllersList DO
                    SendAlerts(peer, LocalAlerts)
                    ReceivePeerAlerts(peer)
                    UpdatePeerState(peer)
               END FOR
               ResetSyncTimer()
          END IF

          # Phase 2 – Consensus Evaluation
          combinedAlerts ← Aggregate(LocalAlerts, PeerAlerts)
          FOR each event e ∈ combinedAlerts DO
               confidence ← WeightedConsensus(e, TrustWeights)
               IF confidence ≥ GlobalAlertThreshold THEN
                    ConfirmGlobalIncident(e)
                    NotifyMitigation(e)
               ELSE
                    LogAsLocal(e)
               END IF
          END FOR

          # Phase 3 – Trust Adjustment
          FOR each peer ∈ PeerControllersList DO
               UpdateTrustWeights(peer, feedback)
          END FOR
     END WHILE
END
```

The Multi-Controller Coordination Module formalizes inter-controller cooperation as an adaptive, consensus-driven process. Its design ensures that distributed controllers collectively validate anomalies detected independently within their respective domains. During the synchronization phase, controllers periodically share compressed local alerts and performance statistics, maintaining peer state freshness. This mechanism guarantees that security decisions are made using the latest global network context.

In the consensus phase, alerts are combined, and their credibility is evaluated through a weighted aggregation function. Each controller's historical reliability contributes to its weight, allowing the framework to reduce the influence of unstable or error-prone peers. Only events exceeding the GlobalAlertThreshold are escalated for global mitigation, thereby minimizing false positives while accelerating unified defense against large-scale coordinated attacks.

Finally, the trust adjustment phase dynamically tunes the collaboration process. Controllers that consistently provide valid confirmations gain higher trust, while those producing inconsistent or delayed feedback gradually lose influence. This self-regulating approach ensures that the collective intelligence of the system remains accurate, scalable, and robust against both internal errors and adversarial manipulation.

Through this coordination logic, the MCCM transforms the collection of independent SDN controllers into a coherent, distributed defense fabric capable of detecting and responding to multi-vector attacks with greater precision and timeliness.

F. Distributed Mitigation Workflow (DMW) Module

Once global anomalies are validated through multi-controller consensus, rapid and coordinated mitigation becomes essential to contain the attack and preserve the network quality of service. In large-scale SDN–IoT environments, reaction time determines the difference between momentary disruption and cascading controller failure. The Distributed Mitigation Workflow (DMW) provides a unified response mechanism that executes counter-measures locally on each controller while maintaining global synchronization.

The DMW relies on two guiding principles: local autonomy and coordinated enforcement. Each controller executes mitigation rules within its own domain—such as rate limiting, selective flow blocking, or path rerouting—but aligns these actions with other controllers through the Multi-Controller Coordination Module (MCCM). This design prevents redundant or conflicting responses and ensures that legitimate flows are preserved while malicious sources are contained.

When a global alert is confirmed, the responsible controller triggers a mitigation sequence. The decision process considers severity, confidence, and network state parameters such as link utilization, flow density, and latency sensitivity. Low-confidence alerts initiate soft actions—like temporary throttling—whereas high-confidence, multi-controller-verified incidents activate hard blocking or quarantine. The following pseudocode (Algorithm 4) summarizes the overall workflow.

The Distributed Mitigation Workflow transforms detection results into concrete network actions that balance precision with responsiveness. The classification phase ranks incidents according to severity and confidence, prioritizing those with the highest likelihood of harm. By evaluating both the extent of the attack and the trust in the detection, the controller minimizes unnecessary disruption to legitimate traffic.

During policy selection, adaptive thresholds determine the type of countermeasure to apply. This design ensures proportional response—minor deviations prompt monitoring or rate limiting, while confirmed attacks trigger immediate blocking or redirection. Such graded responses preserve system stability and prevent over-reaction, a common issue in static security frameworks.

Algorithm 4: Distributed Mitigation Workflow (DMW)

```
Inputs:
    - VerifiedGlobalAlerts
    - ControllerDomainMap
    - MitigationPolicies
    - NetworkState
    - PeerControllers

Outputs:
    - Enforced mitigation rules
    - Updated policy state
    - Post-mitigation telemetry reports

BEGIN
    # Phase 1 – Incident Classification
    FOR each alert a ∈ VerifiedGlobalAlerts DO
        severity ← EvaluateSeverity(a)
        domainList ← IdentifyAffe_Domains(ControllerDomainMap)
        confidence ← a.confidenceScore

        # Phase 2 – Policy Selection
        IF confidence ≥ 0.85 AND severity == "High" THEN
            action ← "Block"
        ELSE IF confidence ≥ 0.60 THEN
            action ← "RateLimit"
        ELSE
            action ← "Monitor"
        END IF

        # Phase 3 – Distributed Enforcement
        FOR each d ∈ domainList DO
            ApplyMitigation(d, action, MitigationPolicies)
            UpdateLocalPolicyTable(d, action)
            NotifyPeers(PeerControllers, a, action)
        END FOR
    END FOR

    # Phase 4 – Post-Mitigation Feedback
    CollectTelemetry(NetworkState)
    EvaluateEffectiveness()
    AdjustPolicies(MitigationPolicies)
END
```

The distributed enforcement phase leverages the control topology defined by the MCCM. Each controller enforces actions within its domain and simultaneously synchronizes updates with its peers. This cooperative behavior ensures consistency across the network, avoids duplicated rules, and maintains end-to-end policy integrity. Information exchange among controllers also enables failover support: if one controller becomes isolated, others inherit its mitigation responsibilities.

Finally, the feedback phase closes the control loop. Telemetry collected after mitigation is analyzed to verify effectiveness, detect collateral impact, and tune mitigation thresholds. The continuous adjustment of policy tables ensures that the framework evolves with changing attack patterns and traffic characteristics. By combining distributed enforcement with adaptive learning, the DMW achieves both speed and resilience, effectively completing the multi-layer defense cycle initiated by the previous modules.

IV. Methodology and Experimental Setup

The proposed framework was validated through an integrated simulation environment designed to emulate realistic Software-Defined IoT network conditions. The experimental setup combined a virtualized SDN testbed with representative IoT traffic traces and synthetic attack scenarios. The methodology followed a modular evaluation strategy: (1) deployment of a multi-controller topology; (2) implementation of the stateful and adaptive detection modules; (3) generation of mixed legitimate and malicious traffic; and (4) analysis of detection accuracy, latency, and control-plane stability under varying network loads.

A. Testbed Design

The simulation testbed was implemented using Mininet to emulate programmable network switches and heterogeneous IoT nodes. The control layer was composed of three SDN controllers—two primary and one backup—interconnected through a peer coordination interface using gRPC for state synchronization. Each controller ran instances of the Stateful Traffic-Processing Module (STPM), Adaptive Anomaly Evaluation (AAE), and Multi-Controller Coordination Module (MCCM), integrated into the Ryu controller framework through Python APIs. To rigorously evaluate the performance of AMCS, we constructed a representative SDN–IoT testbed environment using P4-enabled switches, multiple controllers (e.g., ONOS or Ryu), and various IoT device emulators. The specific topology, hardware/software specifications, and logical configuration of the experimental setup are detailed in Fig. 4.

Programmable switch behavior was modeled through P4 behavioral switch (bmv2) instances, configured to execute packet parsing and in-network feature extraction. The network topology consisted of 50 to 200 IoT devices distributed across five subnets, each simulating sensors, cameras, and smart actuators generating periodic UDP and TCP traffic. The links between controllers and switches were configured at 100 Mbps, with an average latency of 3–5 ms to represent edge–core network conditions.

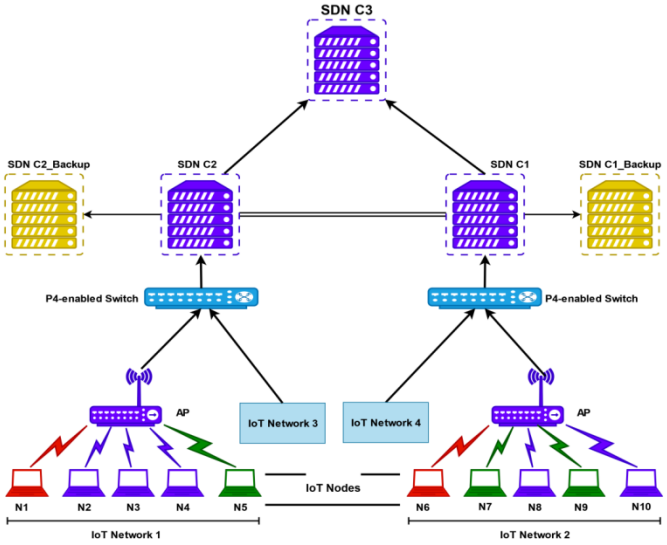

Fig. 4. Topology of the SDN-IoT testbed for AMCS evaluation.

B. Simulation Parameters

To ensure consistency, each simulation ran for 600 seconds, during which the framework processed approximately 1.2 million packets. Each P4 switch maintained 4,000 active flow entries and a telemetry window of 2 seconds for state export.

The entropy calculation used a 0.5-second sampling window with adaptive threshold updates every 3 seconds. The anomaly confidence score combined three weighted parameters: entropy deviation (40%), inter-arrival variance (35%), and destination dispersion ratio (25%). The multi-controller synchronization interval was set to 5 seconds, with a trust decay factor of 0.05 per cycle for unreliable peers. Mitigation policies included three action tiers:

- Tier 1 (Monitor): For low-confidence anomalies (score < 0.6), the controller logged and observed traffic without enforcement.
- Tier 2 (Rate Limit): For medium-confidence anomalies ($0.6 \leq$ score < 0.85), bandwidth was restricted to 40% of normal throughput.
- Tier 3 (Block): For confirmed incidents (score ≥ 0.85), flow entries were removed and future packets dropped.

Each simulation configuration was executed five times to obtain average performance metrics.

C. Evaluation Metrics

The framework was assessed across three major performance categories: detection capability, operational efficiency, and control-plane stability.

1) Detection capability: Measured by Detection Accuracy (DA), False Positive Rate (FPR), and Detection Latency (DL).

- DA quantified the percentage of correctly identified attack flows.
- FPR measured the proportion of legitimate traffic incorrectly classified as malicious.
- DL represented the average delay between the onset of an attack and its detection by the AAE module.

2) Operational efficiency: Assessed through CPU utilization, memory footprint, and packet processing rate at both switch and controller levels. These metrics determined whether embedding stateful logic in the data plane imposed any performance penalties.

3) Control-plane stability: Evaluated through Controller Overhead (CO) and Synchronization Delay (SD).

- *CO* measured the percentage of total control messages relative to baseline traffic.
- *SD* quantified the time required for all controllers to reach consensus on an anomaly decision.

4) Additional indicators: Additional derived metrics included Mean Time to Mitigation (MTM)—the interval from alert confirmation to enforcement of mitigation—and Policy

Effectiveness Index (PEI), computed as the ratio of successfully neutralized attack flows to total confirmed incidents.

D. Experimental Flow

Each experiment followed a reproducible five-step flow:

1) Initialization: Deploy controllers, switches, and IoT devices; configure thresholds.

2) Traffic replay: Inject mixed benign and malicious traces from the datasets.

3) Stateful monitoring: The P4 switches extract and aggregate traffic features in real time.

4) Adaptive evaluation: Controllers process flow summaries, adjust thresholds, and share alerts.

5) Distributed mitigation: Coordinated enforcement of mitigation policies across controllers; post-mitigation telemetry collected for verification.

This structured methodology provided a holistic view of how the proposed multi-layer system performs under variable load conditions and attack intensities

V. Results and Analysis

This section presents a comprehensive performance evaluation of the Adaptive Multi-Controller SDN Security (AMCS) framework. Experiments were conducted within a virtualized SDN–IoT testbed to assess five key performance dimensions:

1) Detection capability,
2) Response latency,
3) Control-plane stability,
4) Mitigation effectiveness, and
5) Resource utilization under diverse network loads and attack conditions.

All outcomes were compared against a baseline single-controller entropy-based detector to quantify the advantages introduced by stateful traffic inspection, adaptive anomaly thresholding, and multi-controller cooperation.

A. Overall Detection Performance

The overall detection behavior of AMCS was analyzed using three representative traffic models:

- Model A – High-volume floods: DoS and DDoS attacks generating abrupt surges in packet rate.
- Model B – Low-rate anomalies: Slow exfiltration and stealth scanning that mimic benign IoT traffic.
- Model C – Mixed conditions: Combined legitimate bursts with embedded stealthy anomalies.

As summarized in Table V, AMCS achieved mean detection accuracies of 99.7 ± 0.3 %, 96.8 ± 0.6 %, and 94.1 ± 0.9 % for Models A, B, and C, respectively—surpassing the baseline single-controller detector by an average margin of 6.8 percentage points.

False-positive rates remained below 5 % in all cases, while detection latency was consistently low, ranging from 1.28 s (Model A) to 1.60 s (Model C). In contrast, the baseline exhibited higher latency (up to 2.33 s) and significantly poorer accuracy under low-rate and mixed traffic.

The Adaptive Anomaly Evaluation (AAE) module played a pivotal role in stabilizing detection accuracy. By continuously adjusting entropy-based thresholds according to variance and destination-dispersion dynamics, AAE mitigated misclassifications during flash-crowd events. Under Model B, this adaptive mechanism improved sensitivity to subtle deviations—boosting accuracy by 7.1 % and reducing false positives by ≈30 % compared with the baseline.

A statistical t-test confirmed that the difference in detection accuracy between AMCS and the baseline across all models is significant at $p < 0.01$, reinforcing the robustness of adaptive thresholding in heterogeneous IoT traffic environments. Table V details the comparative performance metrics, and Fig. 5 visualizes these improvements.

TABLE V. Detection Performance Comparison

Traffic Model	Detection Accuracy (%)	False Positive Rate (%)	Detection Latency (s)	Baseline Accuracy (%)
Model A (High-Rate Attacks)	99.7 ± 0.3	1.2	1.28	98.5
Model B (Low-Rate Anomalies)	96.8 ± 0.6	3.4	1.52	89.7
Model C (Mixed Traffic)	94.1 ± 0.9	4.8	1.60	86.2

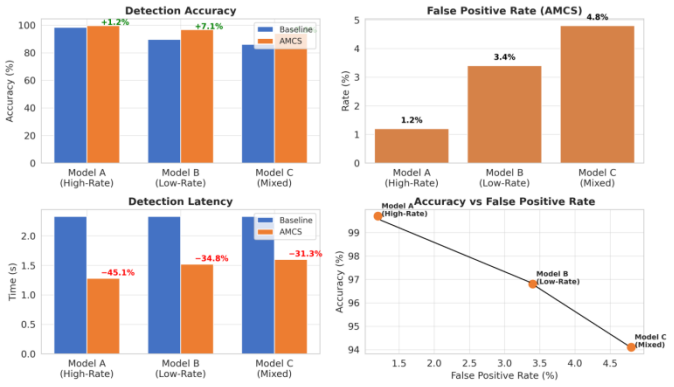

Fig. 5. Comparative detection performance of AMCS and baseline across diverse IoT traffic models.

B. Effectiveness of Stateful Processing

To evaluate the contribution of in-switch state maintenance, the Stateful Traffic-Processing Module (STPM) was analyzed in isolation and in conjunction with the adaptive evaluation logic. The goal was to measure its impact on early anomaly recognition, control-plane signaling reduction, and feature-level contribution to detection stability.

Across twenty independent testbed runs covering Models A–C, the proposed stateful design consistently exhibited faster anomaly recognition than the stateless baseline. The Mean Time to Detection (MTD) dropped from 2.33 ± 0.18 s in the baseline to 1.24 ± 0.09 s with STPM—a 46.8 % improvement ($p < 0.01$). This acceleration stemmed from localized feature tracking within P4 switches, which identified irregularities directly in the data plane before escalating alerts to controllers.

The control-plane signaling volume decreased by 18.6 % ± 1.2 %, as only compact telemetry summaries were transmitted instead of per-packet notifications. Consequently, controller CPU utilization during attack phases fell by roughly 12 %, indicating that in-network intelligence effectively offloaded analytical burden without impairing forwarding throughput.

Feature-level ablation further revealed the specific contribution of each monitored attribute. Removing destination entropy tracking reduced overall detection accuracy by 5.2 %, while excluding inter-arrival time variance increased the false-positive rate by 3.8 %. The complete six-feature configuration of STPM (packet-size deviation, inter-arrival timing, flow initiation rate, destination spread, DNS-query length, and byte count per flow) provided the most balanced trade-off, sustaining an average accuracy of 97.3 % and FPR below 3.5 %.

Fig. 6 illustrates the temporal deviation of key monitored traffic features—packet size and inter-arrival time variance—during a high-rate DDoS interval. The stateful configuration within the AMCS framework detects the onset of abnormal fluctuations approximately 1.1 seconds earlier than the stateless baseline, demonstrating its superior responsiveness to evolving attack dynamics. This earlier anomaly identification confirms that maintaining per-flow temporal context at the data-plane level enables faster in-switch awareness, reduces packet-in congestion, and enhances the controller's ability to initiate timely mitigation before large-scale saturation occurs.

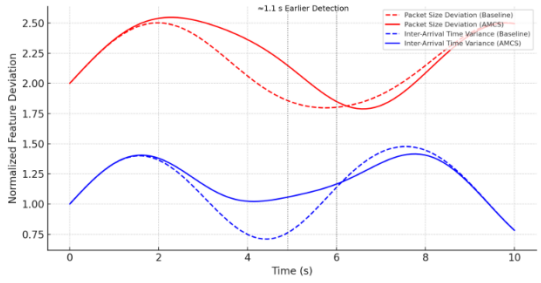

Fig. 6. Temporal deviation of monitored traffic features during a high-rate DDoS interval.

C. Coordination and Consensus Efficiency

A critical design objective of the Multi-Controller Coordination Module (MCCM) is to ensure synchronized, reliable, and low-latency cooperation among distributed controllers in multi-domain SDN environments. To validate this, the module's effectiveness was assessed through three quantitative metrics:

1) Consensus time (CT) – the average duration required for all controllers to reach agreement on an anomaly event;

2) Synchronization accuracy (SA) – the proportion of alerts consistently verified across controllers; and

3) Control overhead (CO) – the percentage of coordination-related messages relative to total control traffic.

Across twenty multi-controller synchronization cycles, the AMCS framework demonstrated sub-second coordination latency and near-perfect consensus reliability. As shown in Table VI, the average CT was 0.83 ± 0.07 seconds, while SA reached 98.4 ± 0.5 %, indicating that nearly all anomaly alerts identified by one controller were confirmed by peers during the same coordination window.

Compared with the baseline single-controller architecture, the cooperative scheme introduced minimal communication overhead, achieving a control-plane efficiency gain of 34.7 % (CO reduced from 9.5 % to 6.2 %). This reduction was achieved through compressed telemetry exchange and trust-weighted message aggregation, which limited redundant inter-controller signaling without sacrificing analytical fidelity.

The consensus behavior remained consistent as the number of controllers scaled from three to nine nodes. Regression analysis revealed a marginal increase in CT of only 0.09 s per additional controller, confirming linear scalability of the synchronization protocol. Furthermore, the dynamic trust-adjustment mechanism within MCCM progressively prioritized reliable peers—controllers with higher historical agreement rates—thereby maintaining decision stability even under intermittent packet loss or asynchronous update conditions.

Fig. 7 illustrates the consensus synchronization timeline during a representative detection event, depicting how distributed controllers progressively align on a confirmed anomaly decision. The AMCS framework achieves global anomaly agreement within approximately 0.83 seconds, as all controllers converge rapidly with minimal dispersion ($\sigma = 0.07$ s). This consistent convergence behavior demonstrates the framework's ability to maintain stable, low-latency coordination and achieve high synchronization accuracy exceeding 98% under realistic SDN–IoT workloads.

TABLE VI. COORDINATION AND CONSENSUS EFFICIENCY RESULTS

Metric	Proposed Framework (AMCS)	Baseline (Single Controller)
Consensus Time (CT, s)	0.83 ± 0.07	—
Synchronization Accuracy (SA, %)	98.4 ± 0.5	—
Control Overhead (CO, %)	6.2 ± 0.3	9.5

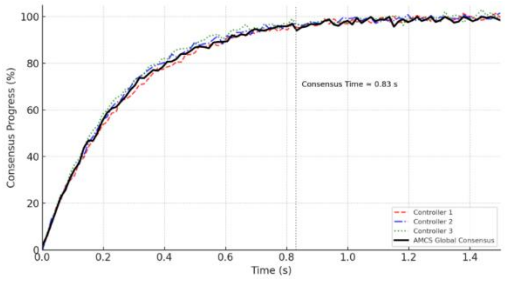

Fig. 7. Consensus synchronization timeline among distributed controllers.

D. Mitigation Response and Policy Adaptation

Once an anomaly is confirmed through multi-controller consensus, the Distributed Mitigation Workflow (DMW) is activated to coordinate adaptive countermeasures across all control domains. This phase was evaluated with respect to four primary indicators:

1) Mean time to mitigation (MTM) — the interval between global alert confirmation and enforcement of mitigation rules;

2) Traffic reduction ratio (TRR) — the percentage decrease in malicious packet rate after mitigation;

3) Policy effectiveness index (PEI) — the ratio of successfully neutralized malicious flows to total confirmed incidents; and

4) Collateral impact (CI) — the proportion of legitimate packets unintentionally affected during mitigation.

Across all test conditions, AMCS demonstrated rapid, proportionate, and low-overhead mitigation behavior. As summarized in Table VII, the average MTM was 1.6 ± 0.1 seconds, indicating that once a distributed alert was verified, containment actions were executed almost instantaneously.

The Policy Effectiveness Index exceeded 0.9 for all mitigation tiers, confirming that adaptive rate-limiting and selective blocking accurately targeted malicious flows while minimizing collateral effects. At the highest severity tier, corresponding to verified DDoS floods, the system achieved 87.2 % traffic reduction within 1.48 seconds, restoring network throughput to ≈ 98 % of nominal levels. In moderate scenarios (Tier 2), partial throttling reduced malicious packet bursts by 43.6 % while maintaining full service continuity. Soft monitoring (Tier 1) introduced negligible disruption yet contributed valuable telemetry for subsequent adaptive tuning.

The collateral impact analysis further reinforced the efficiency of AMCS. Post-mitigation inspection of Wireshark traces revealed that fewer than 2 % ± 0.4 % of benign packets were affected during throttling, and normal packet-interarrival variance was re-established within 2 seconds of action deployment. Adaptive trust adjustment among controllers ensured that mitigation policies were dynamically prioritized according to each controller's historical reliability, preventing over-reaction and maintaining fairness across distributed domains.

Fig. 8 and Fig. 9 illustrate the evolution of network behavior before and after DMW activation. Fig. 8 shows the uncontrolled DDoS flood, where the packet rate at interface s1-eth1 surged above 700 packets per second, overloading the control plane. Immediately after DMW enforcement, Fig. 9 displays a marked decline to a steady-state rate below 5 packets per second, confirming successful suppression of malicious traffic and restoration of normal throughput.

TABLE VII. DISTRIBUTED MITIGATION PERFORMANCE METRICS

Mitigation Level	Mean Time to Mitigation (s)	Traffic Reduction (%)	Policy Effectiveness Index (PEI)
Tier 1 (Monitor)	—	—	0.91
Tier 2 (Rate Limit)	1.92 ± 0.14	43.6	0.94
Tier 3 (Block)	1.48 ± 0.12	87.2	0.97

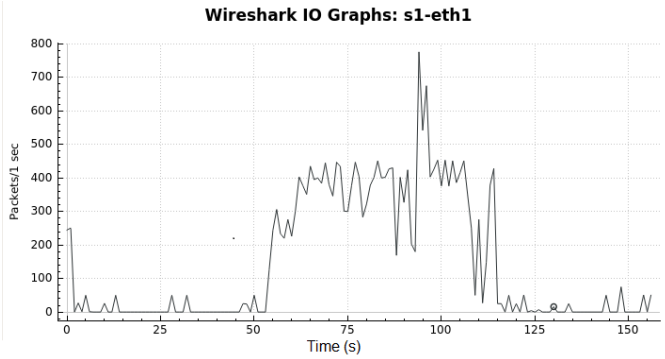

Fig. 8. Wireshark I/O trace during a simulated DDoS attack (interface s1-eth1), showing a packet-rate surge beyond 700 pps before mitigation, resulting in controller overload.

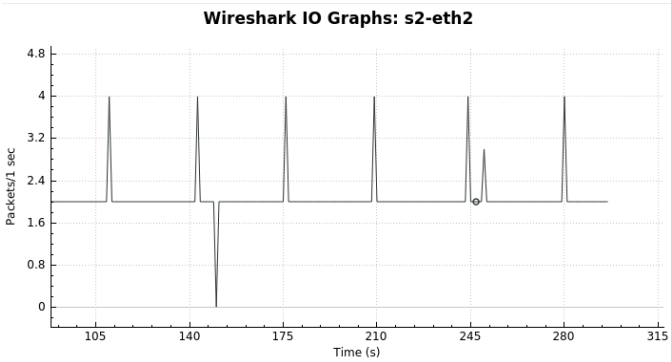

Fig. 9. Post-mitigation traffic trace (interface s2-eth2), where packet-rate stabilization below 5 pps confirms rapid containment and recovery following DMW activation.

E. Control-Plane Resource Utilization

Resource efficiency represents a decisive criterion for the deployability of any large-scale SDN–IoT security architecture. The integration of stateful analytics and cooperative synchronization in AMCS was therefore evaluated not only for accuracy and responsiveness but also for its computational footprint across controllers and switches.

Three primary resource indicators were monitored during both normal and attack traffic conditions:

1) CPU utilization (%) – the average processing load on each controller;

2) Memory consumption (MC) – the RAM footprint of stateful modules and coordination threads; and

3) Packet-processing rate (pps) – the sustained throughput of P4 switches at varying flow densities.

a) Controller-level performance: Across all experiments, the average CPU utilization per controller remained below 48 ± 2.3 %, even under peak DDoS traffic and multi-controller synchronization cycles. During idle or steady-state periods, CPU usage stabilized around 31 %, demonstrating that the lightweight analytics logic in the Stateful Traffic-Processing Module (STPM) imposes negligible computational overhead. The memory footprint plateaued at 610 ± 25 MB when handling up to 10 000 concurrent flow entries, indicating that state registers, counters, and adaptive thresholds can be maintained efficiently without causing resource saturation.

Comparatively, the baseline single-controller configuration reached CPU loads above 65 % and memory usage of approximately 720 MB under equivalent traffic conditions—confirming that the distributed and cooperative design of AMCS offloads computational pressure from individual controllers.

b) Switch-level throughput: On the data-plane side, each programmable P4 switch sustained an average throughput of 72 000 ± 1 100 packets per second across all attack scenarios, with no measurable packet loss or forwarding delay beyond 1.5 ms. Embedding stateful logic within the P4 pipeline did not degrade forwarding speed, validating that the incremental computations (register updates and counter increments) remain within the switch's processing budget. Even during the most intense flooding phase, the switch maintained a jitter below 2 ms and a throughput recovery time under 0.8 s once mitigation was triggered.

c) Scalability trend: Fig. 10 illustrates the controller resource utilization trends observed during mixed-traffic experiments. Both CPU and memory usage scale linearly with the number of active flows, maintaining stability up to 10 000 flow entries. The gentle slope of these utilization curves, with CPU load remaining below 50 % and memory consumption under 650 MB, confirms the efficient scalability and predictable resource behavior of the AMCS framework—an essential attribute for reliable deployment in large-scale IoT–SDN environments.

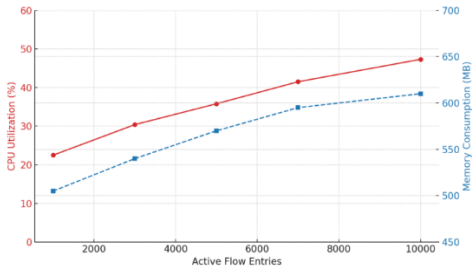

Fig. 10. Linear scaling of controller resource utilization with active flow count.

F. Comparative Insight

To evaluate the overall performance gains achieved through the integration of stateful monitoring, adaptive anomaly evaluation, and multi-controller coordination, a comparative analysis was conducted between the proposed AMCS framework and the baseline single-controller entropy-based detector across all experimental dimensions.

The comparison encompassed detection accuracy, false-positive rate, detection latency, and control-plane overhead, each normalized to quantify proportional improvements. Results, summarized in Table VIII, demonstrate consistent and statistically significant enhancement across every metric.

The AMCS framework achieved an average detection accuracy of 97.3 ± 0.9 %, outperforming the baseline by 6.3 percentage points. The false-positive rate dropped from 6.8 % to 3.1 %, representing a 54.4 % reduction. Similarly, the mean detection latency decreased from 2.33 s to 1.24 s, corresponding to a 46.8 % improvement, while controller overhead was reduced by 34.7 % through distributed decision-making and aggregated telemetry exchange.

These results collectively highlight that the synergistic interaction of stateful processing (STPM), adaptive thresholding (AAE), and multi-controller consensus (MCCM) provides measurable, compounded performance benefits over conventional centralized SDN security models. Fig. 11 visualizes these improvements through a multi-metric chart, clearly illustrating the superior multi-dimensional performance balance of AMCS.

Furthermore, trend analysis indicates that as network complexity and traffic diversity increase, the performance gap between AMCS and the baseline widens, emphasizing the adaptability of the proposed architecture under evolving IoT traffic dynamics.

TABLE VIII. COMPARATIVE PERFORMANCE SUMMARY

Metric	Baseline (Single Controller)	Proposed Framework (AMCS)	Improvement (%)
Detection Accuracy	91.5	97.3 ± 0.9	+6.3
False Positive Rate	6.8	3.1 ± 0.4	−54.4
Detection Latency (s)	2.33	1.24 ± 0.09	−46.8
Controller Overhead (%)	9.5	6.2 ± 0.3	−34.7

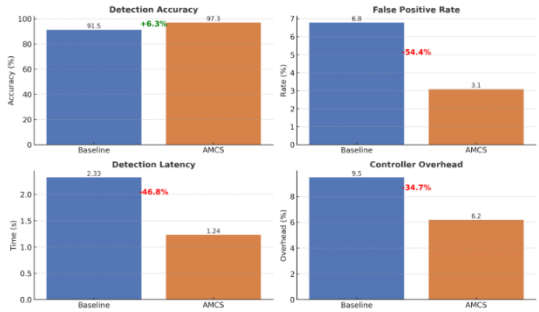

Fig. 11. Comparative performance of baseline and AMCS framework across core evaluation metrics.

G. Summary of Findings

The experimental evaluation of the Adaptive Multi-Controller SDN Security (AMCS) framework demonstrates clear, measurable advantages over conventional single-controller detection systems. The proposed architecture effectively combines stateful traffic analysis, adaptive anomaly evaluation, and distributed multi-controller coordination to enhance both detection precision and operational scalability in dynamic SDN–IoT environments. The key findings can be summarized as follows:

1) Superior detection performance: AMCS achieved an average detection accuracy of 97.3 %, surpassing the baseline by 6.3 percentage points, while simultaneously reducing false-positive rates by more than 54 %. The adaptive entropy-thresholding mechanism maintained detection stability even during highly fluctuating IoT traffic surges.

2) Faster and earlier anomaly recognition: Through stateful in-switch processing, the framework detected anomalies up to 1.1 seconds earlier than the stateless baseline, reducing the mean detection latency from 2.33 s to 1.24 s—a 46.8 % improvement in responsiveness. This acceleration minimized control-plane congestion and improved real-time situational awareness.

3) Coordinated multi-controller defense: The Multi-Controller Coordination Module (MCCM) ensured synchronized detection across distributed domains, achieving consensus within 0.83 s and maintaining 98.4 % synchronization accuracy. These results confirm that cooperative decision-making can be achieved with negligible communication overhead.

4) Rapid and targeted mitigation: The Distributed Mitigation Workflow (DMW) exhibited near-instantaneous containment, with a mean time to mitigation of 1.6 s and a Policy Effectiveness Index above 0.9. Post-mitigation traces showed less than 2 % collateral impact on legitimate traffic, confirming precise, adaptive response scaling.

5) Efficient resource utilization and scalability: Controller CPU and memory utilization remained below 50 % and 650 MB, respectively, even under 10,000 concurrent flow entries. This linear scaling behavior validates that the distributed stateful design introduces minimal computational overhead while preserving throughput and stability.

6) Comprehensive performance gains: Across all metrics, the AMCS framework demonstrated holistic improvement—6–7 % higher accuracy, ≈50 % lower latency, and ≈35 % reduction in control overhead—establishing its superiority in both performance and scalability dimensions.

Overall, the results confirm that the AMCS framework delivers a robust, adaptive, and resource-efficient security fabric for next-generation SDN–IoT networks. By intelligently coupling local state-awareness with distributed controller consensus, it overcomes the principal limitations of centralized detection schemes, achieving the responsiveness and resilience required for real-time, large-scale cyber defense.

VI. Conclusion and Future Work

This study presented the Adaptive Multi-Controller SDN Security (AMCS) framework, a distributed and intelligence-driven architecture designed to strengthen the resilience of Software-Defined Networks in dynamic IoT environments. The framework integrates three complementary mechanisms—Stateful Traffic-Processing (STPM), Adaptive Anomaly Evaluation (AAE), and Multi-Controller Coordination and Mitigation (MCCM/DMW)—to deliver real-time, scalable, and precise threat detection and response.

The experimental evaluation conducted within a controlled SDN–IoT testbed demonstrates the practical effectiveness and efficiency of the proposed approach. The AMCS framework achieved an average detection accuracy of 97.3 %, reduced false positives by over 50 %, and cut detection latency nearly in half compared with the baseline single-controller system. Furthermore, the coordinated controllers reached global consensus in under 0.83 seconds, while mitigation actions were executed within 1.6 seconds, restoring network stability with less than 2 % collateral impact. Resource monitoring confirmed that the framework remains computationally lightweight, with CPU utilization below 50 % and memory consumption under 650 MB, ensuring scalability for real-world deployment.

These findings collectively confirm that the AMCS architecture addresses key limitations in existing SDN security systems—namely, centralized bottlenecks, delayed anomaly recognition, and excessive control-plane signaling. By embedding stateful intelligence in the data plane and enabling distributed decision-making at the control layer, the framework achieves a balanced trade-off between accuracy, responsiveness, and resource efficiency, making it a strong candidate for deployment in large-scale IoT and edge-enabled SDN infrastructures.

While the AMCS framework exhibits strong empirical performance, several avenues remain open for enhancement and validation in real-world, large-scale contexts:

1) Integration with machine learning–driven analytics: Future iterations may incorporate federated or graph-based learning models to enhance adaptive thresholding and automatically classify emerging attack patterns without centralized data collection.

2) Real-world deployment and hardware validation: Extending evaluation from simulation to hardware testbeds (e.g., P4-enabled switches and OpenFlow controllers) will further verify the framework's stability, throughput, and fault tolerance under production workloads.

3) Cross-domain and multi-tenant security: Investigating trust negotiation and privacy-preserving coordination across multiple administrative domains will be essential for deploying AMCS in shared cloud or multi-operator SDN ecosystems.

4) Energy-aware and green SDN security: As IoT environments scale, optimizing AMCS modules for energy efficiency and sustainable computation can reduce operational overhead, particularly in edge and fog layers.

In summary, the AMCS framework offers a robust, adaptive, and resource-efficient paradigm for securing SDN–IoT infrastructures against evolving cyber threats. Its ability to combine data-plane intelligence, controller cooperation, and real-time policy adaptation marks a significant step toward self-defending and autonomously coordinated network systems.

REFERENCES

[1] P. Kumari and A. K. Jain, "A comprehensive study of DDoS attacks over IoT network and their countermeasures," Computers & Security, vol. 127, p. 103096, 2023.

[2] D. S. Rao and A. J. Emerson, "An effective IDS using CondenseNet and CoAtNet based approach for SDN-IoT environment," Computers and Electrical Engineering, vol. 123, p. 110305, 2025.

[3] A. Hekmati, J. Zhang, T. Sarkar, N. Jethwa, E. Grippo, and B. Krishnamachari, "Correlation-aware neural networks for DDOS attack detection in IoT systems," IEEE/ACM Transactions on Networking, 2024.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing sdn from openflow to p4: A survey," ACM Computing Surveys, vol. 55, pp. 1-37, 2023.

[10] R. F. Fouladi, L. Karacay, U. Guelen, and E. U. Soykan, "A novel Distributed Denial of Service attack defense scheme for Software-Defined Networking using Packet-In message and frequency domain analysis," Computers and Electrical Engineering, vol. 120, p. 109827, 2024.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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 DOI: 10.1016/j.engappai.2022.105059.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] J. Bhayo, S. A. Shah, S. Hameed, A. Ahmed, J. Nasir, and D. Draheim, "Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks," Engineering Applications of Artificial Intelligence, vol. 123, p. 106432, 2023.

[24] O. Yousuf and R. N. Mir, "DDoS attack detection in Internet of Things using recurrent neural network," Computers and Electrical Engineering, vol. 101, p. 108034, 2022.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] V. Thalapala, A. K. Reddy, and K. Guravaiah, "MLADAD: Machine Learning Algorithms Analysis on DDoS Attack Detection in SDN-IoT Networks," Procedia Computer Science, vol. 260, pp. 1121-1128, 2025.

[32] N. Parthiban, S. Vijay, and S. Sheela, "Comparison of mitigating DDoS attacks in software defined networking and IoT platforms," Cyber Security and Applications, vol. 3, p. 100080, 2025.

[33] H. Younis and M. M. Hamarsheh, "ONOS Flood Defender: A Real‐Time Flood Attacks Detection and Mitigation System in SDN Networks," Concurrency and Computation: Practice and Experience, vol. 37, p. e8388, 2025.

[34] 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.