International Journal of Sustainability and Innovation in Engineering
(IJSIE)
https://www.doi.org/10.56830/IJSIE2024

Vol.2 2024

Low-Latency DDoS Mitigation: Arista DANZ vs. Cisco Tetration

Ashutosh Chandra Jha
Network Security Engineer, NewYork, USA
Author Email: ashutoshjhany@gmail.com

Received: 2 August 2024. Accepted: 11 October 2024. Published: 10 December 2024

Abstract

This study assesses the performance of low-latency protection against DDoS attacks in contemporary leaf-spine and hybrid-cloud datacenters, where queuing delays of milliseconds can violate SLO and broadcast attacks. It introduces a side-by-side approach to compare Arista DANZ and its use of DANZ Monitoring Fabric, sFlow/IPFIX, ERSPAN, and gNMI with Cisco Tetration (Secure Workload), which measures hosts using kernel-level sensors, and micro-segmentation can be enforced. The cure-to-end is examined: from the first anomalous telemetry window to the first verified drop/redirect. A more controlled testbed is used with PTP-synchronized hardware timestamps, synthetic (TRex/MoonGen) and replay first-trace replay, export scaling of 100-500 ms, and 1:512-1:2048 sampling. The features are comprised of SYN/ACK divergence, flow/packet rates, inter-arrival variance, source/port entropy, and five-tuple fan-in/out. Performance metrics have been detection time, enforcement time, p99-p99.9 one-way and overall-delay deltas, false-positive and false-negative rates, TCAM occupancy, controller/collector CPU and API throughput; verification has been coupled with mirrored packet captures and device counter information and policy acknowledgments. Fabric-first mitigation reduces time-to-mitigate in volumetric L3/L4 floods through ACLs, policers, BGP Flowspec or RTBH, whereas host-centric enforcement is more effective in app-layer low-and-slow traffic and intra-VLAN flood jumping; a hybrid trigger path offers the strongest blast-radius shortening and accuracy. Contributions encompass a vendor-neutral control-loop budget, a repeatable harness and failure-injection regimen, operational SLOs and rollback playbooks, and practical recommendations on sampling and export cadence and rule-churn limits; deployment recommendations are delivered. The scope is enterprise datacenters and hybrid clouds; forensics and external scrubbing will remain out of scope at the moment.

Keywords: Low-latency DDoS mitigation, Arista DANZ (DANZ Monitoring Fabric), Cisco Tetration (Secure Workload), Streaming telemetry (sFlow/IPFIX/NetFlow, gNMI), BGP Flowspec (with RTBH).

1. Introduction

Development of distributed denial of service attacks has come a long way since the simpler bandwidth floods. Attackers have now dissected the volumetric floods like UDP amplification attacks, which go along with protocol abuse attacks like SYN or ACK flooding, and with application-layer attacks that leverage the processing capabilities of the servers via slow HTTP or HTTPS connections. These types of attacks are commonly instigated in blended campaigns, which make detection and mitigation especially difficult. Volumetric floods can crash the links and buffers within a few seconds. Protocol attacks find vulnerabilities in the handshake mechanism and state tables. In contrast, application-layer attacks exert pressure and become persistent, disregarding the traditional volumetric protection mechanisms.

High latencies specifically affect modern data centers and cloud platforms due to the latency thresholds required by real-time services. Applications like financial trading systems, video, and distributed storage applications are those that depend on deterministic microsecond-level latency. Microbursts of congestion can cause tail latency spikes that may exceed service level goals and ultimately service level agreements. West-East data center traffic is especially susceptible to spikes in latency as services typically have hundreds of micro transactions per second. Any mitigation solution that can add a measurable amount of delay may outweigh the benefits of the attack itself by making your users' work and/or workflows unusable. Hence, the importance of low-latency mitigation as a desirable property becomes a necessity in such an enterprise and cloud environment.

The problem here is that in most situations, the security controls pay attention to accuracy and coverage without a close view of detection to enforcement latency. Intrusion detection systems and firewalls have traditionally been tested in terms of throughput and detection accuracy, with little thought put into quantifying the hidden delay between anomaly detection and execution of a rule. This gap is critical at the time of high-speed spine leaf networks and hyperscale workloads. An architecture whose identification capability is fast, but the enforcement is only several seconds slow, will permit millions of malicious packets to obtain passage without mitigation. On the other hand, an excessively vigilant mitigation system can block good traffic, thus resulting in costly false positives. Both the detection pipeline and the enforcement pipeline must be carefully evaluated to balance those requirements.

This study is motivated by the lack of vendor-neutral assessments that can assess the complete detection-to-enforcement loop. Current studies further fall short by testing the accuracy of anomaly detection in isolation, and even industry marketing documentation may point out throughput figures and not set latency budgets to achieve policy enforcement. Practitioners are deprived of consistent criteria for selecting between solutions when they work at different layers of the network. Rista DANZ Monitoring Fabric delivers fabric-level visibility into telemetry and flow directly to Cisco Tetration. They both state that expert will provide low-latency mitigation, but offer limited direct comparison under standard conditions.

The contribution of this article is to provide a test harness with reproducible results on detection latency, policy enforcement latency, and operational overhead in an equally comparable manner. The assessment measures a vendor-independent latency budget and establishes quantifiable performance targets via both synthetic and replay-based attack traffic. The study instead uses practical metrics, including API to ACL installation time, telemetry export intervals, sampling error rates, and CPU or TCAM utilization, so that the information can be used by the network engineers and security architects directly in making design and deployment choices.

This research is organized into three fundamental objectives. One of the objectives is to explore how fast telemetry pipelines can detect the beginning of an attack, at realistic sampling rates. This will cover testing the export periods of sFlow and Netflow telemetry, the latency of the streaming telemetry, and the capacity of the host agents to report anomalies at scale without overwhelming the control plane. The other objective is to determine the end-to-end time, the move from anomaly detection to enforced blocking or redirection. This metric includes controller processing, API forwarding, and rule insertion in switch TCAMs or host kernels. The last objective researches the accuracy and cost of the detection mechanisms of all the solutions applied in every case. These expenses do involve false favorable rates, false negative rates, resources used, and complications of administration. The study takes on a narrow approach in scope, focusing on aspects of DDoS mitigation in high-performance enterprise networks and data center networks. Long-term forensics, threat attribution, and external scrubbing services are recognized but are not the primary focus.

The study is organized into several chapters, including a literature review, where the latest environment is surveyed to identify gaps in terms of latency-based assessment of DDoS mitigation. Chapter methods and techniques outline the dataset and preprocessing, feature selection, as well as the system for designing the comparative framework. The architecture chapter goes into a detailed analysis of the low-latency telemetry pipeline and enforcement mechanisms that have been implemented in Arista DANZ and Cisco Tetration. The experiments and results chapter provides benchmark results of all systems having realistic attack scenarios and measurement of the detection and enforcement time latency. The discussion chapter makes a qualitative interpretation of these results and identifies operational implications and limitations as well as a forward-looking perspective. The last chapter concludes by summarizing significant findings and making recommendations on deployment strategies.

2. Literature Review

This literature discussion reviews the technical basics of low-latency mitigation of distributed denial of service activity in contemporary leaf-spine and hybrid cloud environments. The requirements are to detect mechanisms that reduce the time span between the onset of an attack and its mitigation, while maintaining service level objectives of throughput and tail latency [27]. The paper starts by giving categories of attack behaviours and the effects on queues and control planes. It then instantiates a control loop Framing whereby the pipeline of mitigation is framed in Phases of senses, appeals to the decision and action under set time budgets. The review then looks at telemetry and analytic methods that can run at wire speed yet be resource aware. It ends with the system archetypes that guide the course of the vendor and gaps that drive the transaction hypothesis in the comparison of hypotheses of Arista DANZ and Cisco Tetration.

2.1 DDoS Taxonomy & Low-Latency Constraints

These volumetric attacks flood links and buffers with high packet rates beyond the capacity of the interface or an internal queue. Reflection/amplification campaigns use open services to scale traffic so that the ingress and upstream buffers fill more quickly than the downstream devices have time to respond. Direct floods also attempt the same result, and they induce constant queuing in egress pipelines. In state exhaustion attacks, the protocol and platform state are attacked with either expensive transitions or a large number of half-constructed connections opened. Illustrative examples are SYN floods, which swell embryonic connection tables, and protocol manipulation, which cause costly control path work [20]. The aim of the application layer attacks is different, and they do not focus on keeping large average rates. Consequently, their values are moderately low, but they have a large number of concurrency or adverse request patterns. Low and slow patterns open many sessions, drain service thread pools, and exceed simple rate limits.

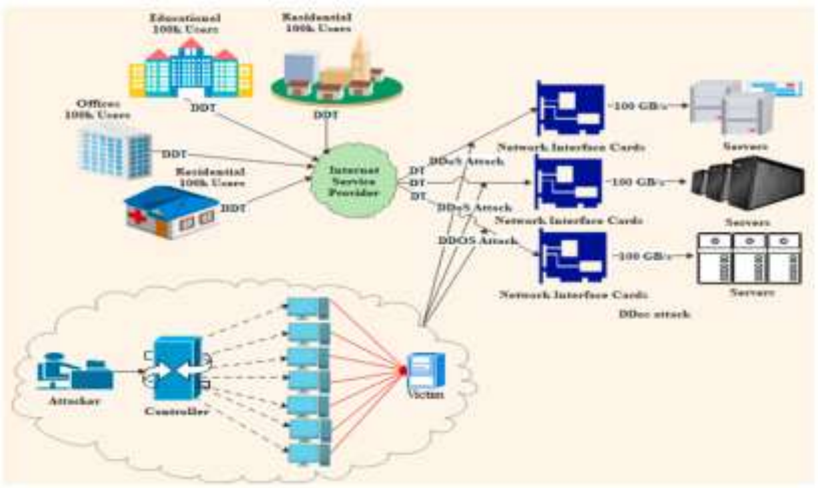

Figure 1: DDoS vectors overwhelming links, buffers, states, and application threads

The low latency requirements arise due to the rapidity of harm growth in the early seconds of an event. The key to tail latency protection of east-west transactions is a response to shallow thresholds of queue depth to prevent backpressure and timeouts. A pragmatic model adopts a closed-loop process where it senses, decides, and acts on mitigation [1]. The packet header flow statistic requires the sense stage to collect the queue depth and service metrics windowed in tallies of hundreds of milliseconds. Suitable playbooks that should be kept within a comparable budget should be chosen at the decision stage by evaluating the anomalies so that queues do not go into unstable areas. The act stage needs to put in place an access control entry diversion path or rate policy within tens of milliseconds at the PI in question. These are the targets of the difference between detection that informs in forensics and detection that prevents collapse. Such performance cannot be achieved without a level of automation discipline that de-emphasises humanity-in-the-loop during the initial response, and continuously validates changes during regular operation, as might be expected in industry-standard secure delivery mechanisms that incorporate automated checks into release pipelines and operational workflows as a means of timely risk reduction [14].

2.2 Telemetry & Analytics Approaches

Packet and flow sampling allow early signal and scale with a bounded device overhead. Under a configured sampling rate and interval, sFlow exports sampled packet headers and interface counters. In the case of a hot segment, the sampling rate can be increased during an incident, but other links can be left at coarser settings in order to limit the export. NetFlow and PFIX record flow-specific details like byte and packet counts, timestamps, and next-hop information on small flat templates that collectors can quickly index. Each interval is used in combination with sampling ratios to define the granularity of detection and collector loads. Short samples and aggressive sampling are sound in offering time to first evidence, but they increase transport and storage requirements. Lengthy buffers minimize overhead and increase the probability that a brief burst does not get detected [31]. The choice of keys and sampling policies becomes pivotal in regulating false deletions during onset as well as false confirmations during microbursts.

Paired sampling with streaming telemetry via gNMI and gRPC provides low-jitter state and counters. Queue depth interface drop pause frames and buffer occupancies inform of the congestion formation before the loss can be distributed to the hosts. Push-based delivery decreases latency between occurrences and their receipt at the controller compared to scrape-based polling, which is vulnerable to schedule jitter and head-of-line blocking. The tradeoff changes to sustained device to collector bandwidth and to backpressure in the ingestion path. The design of scalable communication emphasises asynchronous pipelines (admission control and fan-out patterns) to avoid allowing a single noisy device (or one slow consumer) to reduce visibility fleet-wide, as a message it adds or consumes must reach the majority promptly before it can impact visibility [29]. In practice, operators have implemented message brokers with partitioning to isolate end-to-end throughput and bounded queues with drop policies that prioritise recent data during overloads because stale counters are less valuable to low-latency control.

The signals required to be extracted by the Analytics are robust and need to tolerate sampling error within windows of sub-second intervals. Sketch-based algorithms offer approximate heavy hitters and cardinality estimation in sub-linear memory, which is appropriate to the speed and scale of sparse fabrics. Light sketches can facilitate top-talker detection even without complete tables, and hyperloglog estimates can handle the large number of sources that frequently spike in the reflection and scanning sources. Exponentially weighted moving averages and cumulative sum tests can detect changes in rates and in variances without detailed distributional assumptions. The source and destination entropies contribute to distinguishing between reflection and natural traffic, and port-diversity and traffic-concentration metrics provide amplification fingerprints. At the application layer, network characteristics are combined with service-to-service metrics, which include request rate per identity, latency histograms, and error code distributions. The consolidated perspective helps to minimize dependence on any single indicator and to create a playbook that is targeted at a specific target role, rather than generic actions based on prefixes.

2.3 System Archetypes & Vendor Directions

Two archetypes dominate low-latency mitigation. In the switch-centric archetype, sensing and enforcement reside in the fabric. Switches send sFlow or IPFIX and optionally mirror traffic, using ERSPAN or taps, selected by the operator. A controller consolidates flows, identifies anomalies, and inserts access-control entries, rate limiters, or quality of service policies at ingress. When it comes to interdomain response, the controller pushes BGP Flowspec or prompts the far-off black hole routes to divert the unwanted traffic before it reaches the congested links. This model has strength in that the distance between the decision and enforcement is close, thus no core buffers can clog, and the common links are guarded. Among the main risks are the ternary content addressable memory pressure, rule compilation latency, and the risk of the unrelated feature starvation rule churn. The above risks are addressed through hierarchical templates, rule aging, and guardrails that limit the update rates.

As shown in the figure below, archetype-based mitigation design works through finding determinant system attributes of the DDoS defense system, dimensionality reduction, quantifying similarity between systems, and defining explicit rules that describe two archetypes: switch-centric fabric sensing/enforcement and controller-coordinated interdomain response. Telemetry (sFlow/IPFIX, ERSPAN/taps) is fed to the controller, which synthesizes ingress ACLs, rate-limiters, quality of service, and BGP Flowspec/RTBH. TCAM pressure, compilation latency, and rule-churn starvation risks are mitigated through hierarchical templates, rule aging, guardrails, and ongoing validation against independent observations to provide a rapid slice-safe response.

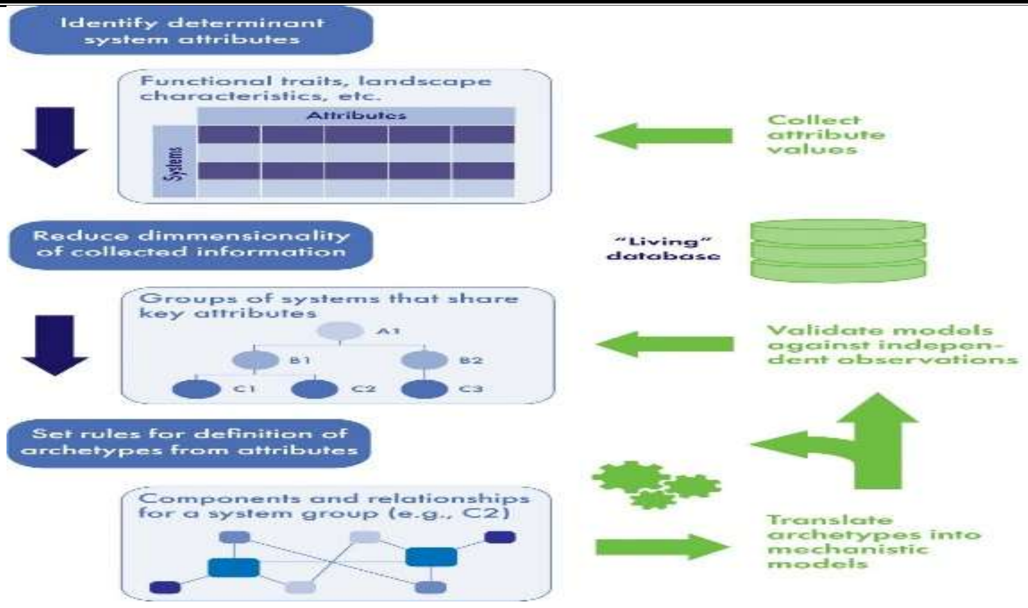
Identify determinant system attributes
Functional traits, landscape characteristics, etc.
Attributes
Systems
Collect attribute values
Reduce dimmensionality of collected information
"Living" database
Groups of systems that share key attributes
A1
B1
B2
C1
C2
C3
Validate models against independent observations
Set rules for definition of archetypes from attributes
Components and relationships for a system group (e.g., C2)
Translate archetypes into mechanistic models

Figure 2: Defining and validating low-latency DDoS mitigation system archetypes

The centric archetypes instruments are host and workloads, as well as hypervisors. Agents track the flow of inventory of the process identity, user identity, and application metadata, and they enforce micro segmentation that restricts east-west traffic. It can be enforced by means of kernel hooks, programmable eBPF, or hosted firewalls based on platform capabilities. Rich context enhances the classification accuracy of application-layer attacks and lateral movement since the system can differentiate between an authorised process and an unanticipated process using the same socket. Costs are in CPU and memory overhead, and in the requirement to distribute policy on a timely basis when thousands of endpoints switch in concert. The history of other areas of feedback systems implies working towards better performance through quick measurement and immediate feedback to reorient direction, also holds in the case of host-centric enforcement, where, through incorrect rules, application behavior can be destabilized unless there is constant feedback to close the loop [11].

The chaining of scrubbing centers can complement the issue of scrubbing centers. Where classification confidence is high and target prefixes are properly scoped, the controller directs traffic to egress capacity that does deep filtering at scale. The first systems to fabricate BGP advertisements or use policy-based routing on the edge direct only the affected prefixes, while all other traffic continues to use the shortest path. First Systems' direct targeted application groups with layer seven safeguards that enforce identity-aware policies and user-based challenges [12]. In both cases, the design has to maintain low latency to unaffected flows and also roll back accepting traffic on the direct path when indicators are restored. Often, the diversion is accompanied by on-path sampling to ensure that the filtered stream generated has the characteristics expected and to capture spillover to reflectors.

2.4 Research Gaps & Hypotheses

There are gaps even with the mature telemetry and enforcement options. Encryption lowers the discriminative power of the packet features and promotes the use of timing and endpoint features. Ternary content-addressable memory budgets, watchlist growth, and rule churn must be managed by switch-centric systems so that mitigations do not block critically important policies and that compilation queues do not hold up immediate installs. Host-centric systems must limit agent overhead, and the calculation and distribution of policies between servers should be able to maintain reasonable bursts of events. Orchestration of cross-plane is frequently split up among fabric controller managers, host managers, and cloud gateway, and adds to end-to-end latency in case of an attack that cuts across domains.

The hypothesis arising out of the control loop framing is clear. The fabric-first approach is likely to reduce time to mitigation during volumetric and state exhaustion attack cases since the sensing and enforcement are performed near the filled queues. A host-centric design will likely be successful in application-layer attacks. It may help to contain lateral movement as flows are observed in context and can be limited prior to entering the fabric [7]. A hybrid combination of fabric mitigation efforts (blast radius reduction) and host-based mitigation efforts (precision) is likely to provide resilient results in mixed workloads. These hypotheses shape contrastive analysis of Arista DANZ and Cisco tetration by addressing the latency of detection, latency of enforcement, and latency of accuracy under sample and operational headroom in churn sustained.

3. Methods and Techniques

3.1 Description of Data Set

This experiment compared low-latency mitigation by constructing a mixed traffic corpus that represents the realistic data center under stress conditions. Benign workloads consisted of gRPC-based microservice calls, transactional database queries, distributed log ingestion, and Kafka streaming that simulates bursty producer-consumer patterns. The application servers produced request fan-ins and fan-outs, and realistic keep-alive settings and retransmission settings. Background traffic was composed of replication traffic and snapshot traffic to ensure that there was traffic traveling east-west on the fabric [9]. The composition guaranteed a plausible baseline performance rate of flows per second, packets per second, short-lived connections, and enduring sessions in a good percentage and mixture to a contemporary service landscape.

Attack traffic included volumetric profiles, such as SYN floods and UDP reflection floods, as well as application-level patterns that affect service but leave bandwidth available. The application layer package incorporates HTTP version two, quick reset, and slowloris-like connection hold with small windows progress. Packet sources consisted of a combination of record traces and high-fidelity generators. Rate control was used to ensure accurate replay of historical packet capture datasets to allow anomalies to be replicated across identical seeds with each run. TRex and MoonGen were used to generate synthetic streams to tune inter-arrival distributions and packet size distributions, and burst structure to millisecond precision. Flow exports were sent to switches and hosts to facilitate cross-validation and to capture flows not mirrored during packet capture. The baseline load was ten to forty gigabits per second, with stress at sixty to one hundred gigabits per second. Per-window and per-flow ground truth labels were based on side channel markers on the generators and five-tuple tags in the replay engine. The corpus was divided into training sets, validation sets, and holdout sets using distinct seeds to avoid leakage across folds and to permit reproducibility.

3.2 Data Preprocessing

Every measurement was based on accurate timing. All the equipment and traffic sources were part of a Precision Time Protocol domain consisting of a hardware grandmaster clock. Timestamps were dragged to the same epoch and aligned to analysis windows with a length of between 100 and 500 milliseconds. The length of the windows was determined by preliminary trial runs that attempted to achieve a good balance between change detection sensitivity and computational load on the collectors and controllers. The data were z-score scaled when distributions were approximately normal and robustly scaled on medians and median absolute deviations when the tails were heavy [10]. Deduplication of packet captures was done at the flow table join by checking sequence numbers to reduce the occurrence of mirror-induced duplicates. Flow logs were reindexed as strict monotonic time series by device to enable evaluation of joins that occur downstream without violating cross-clock skew.

Table 1: Key data preprocessing steps for low-latency DDoS analytics

Stage/Area	**Technique**	**Purpose**	**Key Parameters/Notes**
Time alignment	PTP hardware grandmaster; common epoch; fixed windows	Consistent timing for joins and analytics	100–500 ms windows chosen via trial runs
Feature scaling	z-score or robust (median/MAD)	Normalize features while handling outliers	Choose by distribution shape (normal vs heavy-tailed)

Stage/Area	Technique	Purpose	Key Parameters/Notes
PCAP de-duplication	Flow join with TCP sequence checks	Remove mirror/ERSPAN-induced duplicates	Performed during packet→flow stitching
Flow-log indexing	Strictly monotonic per-device time index	Safe downstream joins without cross-clock skew	Device-scoped series preserves order
Indicator design	Header-only (payload-agnostic) metrics	Work under encryption; privacy aligned	No payload inspection required
Rate/ratio metrics	Packet/flow rates; SYN:ACK on server sockets	Detect volumetric/state-exhaustion onsets	Computed per host and upstream hop
Burst/concentration	Inter-arrival variance; Shannon entropy; fan-in/out	Surface microbursts and amplification/focus patterns	Low entropy → concentration; high fan-out/in → spread
Scale & reliability	HyperLogLog; back-pressure batching; idempotent sinks	Efficient cardinality; stable replays/rollbacks	Approximate uniques; safe re-runs during maintenance

Indicators, which move during denial of service onsets, were designed to guarantee their functionality without inspecting payloads. This objective was to enable their ability to identify traffic, allowing them to serve as furnished-based indicators. Packet rates and flow rates were monitored in terms of the host and per upstream hop, with the ratio of SYN to ACK count on the server sockets. Inter-arrival variance was calculated as a moving statistic to indicate microbursts that do not alter mean rates. Shannon entropy was also computed in distributions over the source addresses and destination ports to identify patterns of amplification, or high-concentration attacks. The derivation of fan out and fan in of the five-tuple was based on different counts of the counterparts per window [17]. HyperLogLog was used in estimating approximate cardinality, exceeding the peak cost of entirely distinct counts. Streaming workers with back-pressure-aware batching and idempotent sinks enabled stream compute jobs to be re-run safely during pipelines under maintenance. This strategy was consistent with the convergence of predictive analytics and operations automation, where repeatable decision loops and controlled rollouts in production networks require durable and versioned data pipelines [16].

3.3 Data Exploration using Visual Analytics

In individual cases of enforcement, exploration-based threshold decisions and implementation of prior choices based on observable behavior preceded attempts at instigating any enforcement experiment. Graphical displays included time-varying percentiles of end-to-end latency, with a primary focus on the 95th and 99th percentiles that determine user experience under load. Log-log plots of exponentially weighted moving averages of flows per second and packets per second were provided to visualize the regime shifts, but downplay the transient noise [6]. Entropy time series indicated the concentration events and the early warning of the reflection waves before the line rate saturated. Candidates that gave rise to anomalies had content that was annotated on charts and linked to runbook entries such that analysts could follow the decision lineage back to evidence.

Spatial and relation patterns were analyzed using the scatter plot and heatmaps that project the source addresses and target ports into short time bins. The identification of concentration ridges was used to identify service abuse, even when aggregate bandwidth was modest. Cumulative distribution functions of per-flow round-trip time deltas before and during the attack window were compared to the baseline to ensure that mitigation did not introduce short diagrams and only an apparent reduction in drop counts. Packet loss and jitter were displayed together with latency percentiles to enable a user judgment about mitigation based on what the user sees, rather than the names of counters. Cross filtering enabled an analyst to click a spike on the entropy panel and see the relating target ports and edge devices. The visualization stack remained off the query system [24]. It preserved the rendered products of the visualization as well as the attached metric queries to make them available to independent auditing and replication.

3.4 Feature Selection and Dimensionality Reduction

An initial reduction of the feature set was achieved through pairwise correlation analysis to avoid redundant signals that increase variance with no discriminative benefit. Absolute correlations above a conservative criterion of near zero point eight five retained the variable by stability under oscillation of baseline rates and lower frequencies of missingness across devices. Principal component analysis was then used as a diagnostic to visualize the concentration of variance and to examine loadings that indicated multicollinearity. The components were not used directly for real-time classification since interpretability and deterministic update cost were given higher priority than compactness. The analysis, however, informed the elimination of weak indicators and affirmed that the engineered indicators explained the majority of the variance that discriminated between the attack windows and benign windows.

Mutual Information scores were calculated between each of the attendants and the labels of attacks over stratified folds to penalize variables based on predictive salience. The ratio of SYN and ACK counts, as well as entropy across source distributions, were among those ranking high as they were sensitive to reflection floods [2]. Variance of inter-arrival times was revealed as a leading indicator of microburst-driven queuing on edge switches. The last feature synchronized the variety with export and compute overhead in collectors and controllers, with consistency of detection performance. The schedules were optimized so that polling and streaming device counters on commodity servers could operate in sub-second scales, and analytic operations could be performed with sub-second budgets per commodity server.

3.5 Comparative Framework Design

A vendor-neutral framework would compare end-to-end loop, telemetry ingest, and enforced mitigation in a repeatable fashion. Detection Time was the time between the first analysis window that passed the configured anomaly threshold and when the anomaly analytics engine generated an actionable alert. Enforcement time is defined as the time between the first accepted policy change on the control channel and the first detected drop or redirect in the data plane, measured with hardware timestamps at line rate [25].To assess user impact, one-way delay deltas between the baseline and attack states and the attack and mitigated states were computed at the ninety-ninth percentile. The number of false positives and negatives was counted over the labeled windows, and the confidence intervals were calculated using bootstrap resampling. Resource headroom was monitored on the two planes. During dynamic mitigation, CPU utilization was measured on the collectors and controllers, and TCAM utilization and rule churn rates were calculated on switches [37]. To evaluate the capacity of controller application program interfaces, the framework would count the number of updates accepted per second and monitor the depth of queues during bursty rule installations.

Repeatability was ensured through fixed seed replays, meaning that each scenario could be re-generated by individual reviewers. The sequence of actions, such as traffic ramp telemetry sampling configuration, threshold warm-up, and policy rollback, was encoded in the runbook to a safe baseline. Control links relied on quality of service markings and the physical independence of paths to avoid bias due to data plane congestion. The framework took into account that analytics collectors and policy engines are more often implemented as a microservice, and cost-scale adjustments have to be made to maintain continuous protection in production settings [3]. Failure injection tests included time synchronization loss, loss of a collector shard, and a temporarily unavailable controller. The safe fallback engaged in each failure mode was such that it allowed minimal policy churn and kept a conservative deny or rate limit posture until telemetry and coordination were present. The complete methodology resulted in similar measurements of detection time, enforcement time, user-facing latency, and resource safety, which are the core factors of low-latency quality mitigation in modern data centers.

4. Low-Latency Telemetry & Enforcement Architecture

4.1 Sensors & Telemetry Pipelines

Reaction to distributed denial-of-service events in subseconds can be reached using disciplined sensing. An Arista DANZ solution with ports exporting either sFlow or IPFIX records directly out of the forwarding ASIC and the DANZ Monitoring Fabric aggregating and enriching these streams to provide real-time analytics. Operators choose five-tuple, ToS, VLAN, and ingress, and they configure sampling on a 1:1-to-1:2048 continuum; export intervals range between 100 and 500 milliseconds depending on granularity and broker capacity. Exporters should use out-of-band links to maintain telemetry in the event of a surge. On high fan-in spines, DANZ nodes de-duplicate, record-coherentize, and create rolling entropies by source IP/port to discover sudden concentration. Adaptive reporting resembles the telematics patterns, which aim to reduce the use of the linkage but maintain the freshness [22].

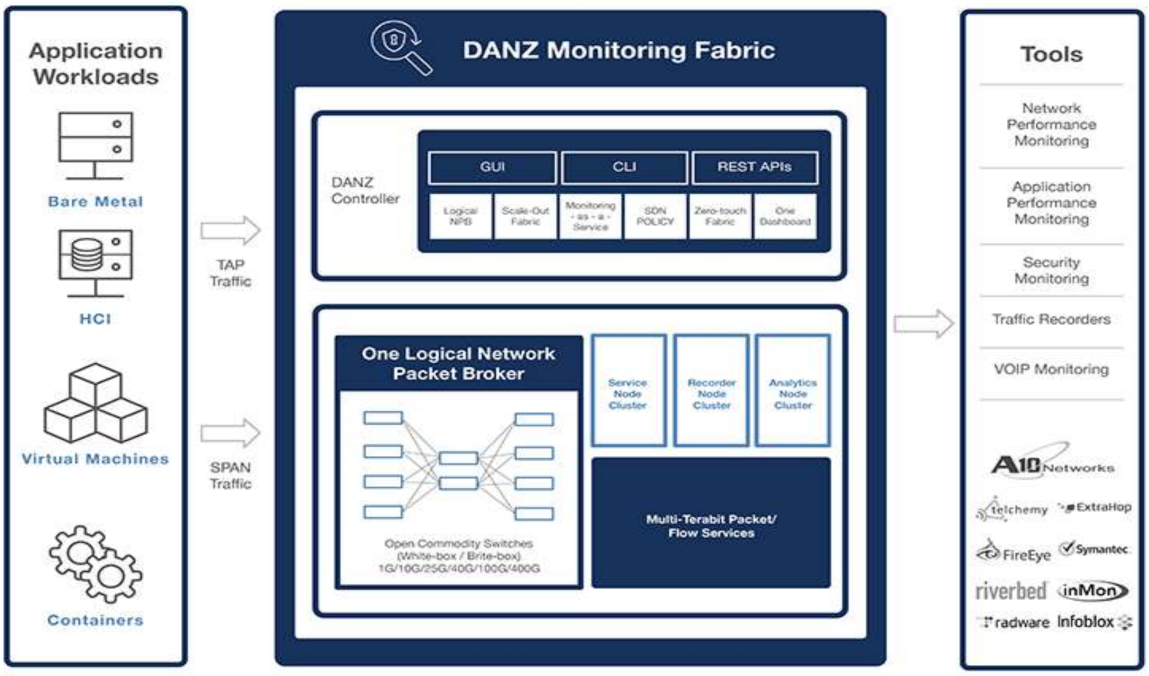

Figure 3: Arista DANZ Monitoring Fabric: subsecond sFlow/IPFIX telemetry and analytics

Cisco Tetration, currently branded as Secure Workload, focuses on the instrumentation of the hosts and workloads. Kernel agents monitor opens, closes, socket metadata, and process identity at the server edge to enrich flow information with context on the user and the package [34]. That perspective facilitates accurate micro-segmentation and attribution of application behavior, especially in the encrypted east-west paths, where deep packet inspection is not practical. Collectors construct time-delimited summaries, map to workload identities, and push their events to a central policy engine. The fact that the agent has direct access to process context means that it can attribute spikes to new binaries, container images, or user sessions, allowing for a selective quarantine instead of a full fabric drop. In DDoS onset, such a perspective reduces the search time of root-cause isolation on application-layer abuse with no real-time payload analysis.

Transmission of telemetry via gNMI augments both methods by transmitting a granular time series of counters, including queue occupancy, tail-drop events, ECN marks, and interface pacing. Since the system pushes changes, the latency of alerts is determined by publish intervals and device back-pressure. Operators generate alarms during watermarks placed at the egress to raise an alarm when buffer depth reaches a few milliseconds of wire time, an early sign of the onset of microbursts or reflection floods. A well-designed pipeline connects flows and counters with stable device identifiers, timestamps with hardware counters, and then writes to a persistent queue so detection does not stop with increasing load. In-band and network telemetry, such as per-hop queue depth and latency variance, are made available where possible to distinguish between path-local and target-local loss.

4.2 Data Plane & Silicon Considerations

The key to low-latency mitigation is the usage of merchant silicon behavior. Line rate operations include parsing, lookup, and action by pipeline stages; deterministic match latencies scale with rule layout and memory type. Ternary content-addressable memory is sparse and power-intensive, and offers flexible wildcard matches, but SRAM is denser and has lower latency with exact matches. Policies that are efficient reduce ternary, cluster on common prefixes, and are ordered with the most selective rules at the front of the table. ACL lookup order should be verified throughout, as should shadowed denies [13]. On hierarchical tables, with coarse-grained denies at the highest level and high-priority exemptions at the lowest level, cross-bar contention and worst-case latency are reduced.

Packet mirroring and ERSPAN need to be specifically rate-controlled. Each mirrored packet uses queue space and headroom; during an attack, uncontrolled mirrors multiply drop processing-congestion on a shared outbound port and add tail latency to productive traffic. One possible practical approach will be to mirror only a hashed subset on hot paths, and to police the egress on the ERSPAN, and more preferably to capture locally on idle links where possible. Ingress and egress hardware timestamping allows ground-truth measurement of insertion delay inserted by the switch, notably when correlated to a PTP grandmaster—Sans synchronization, apparent mitigation times skew, obscuring control-plane regressions that they otherwise would.

The strength of sampling has a powerful effect on detection granularity and the cost of exports. With an intelligence level of 1:1024, small botnets and scans may not be detected immediately, but the export volume is still within the bandwidth limits of trunks linked at 100 GbE; at 1:128, detection windows are still narrow, though collectors and brokers will have to absorb bursts of flow records. A sensible operating program outlines attack profiles with more intense sampling and less extended export periods that automatically decrease when thresholds are exceeded, and then, as stability returns within a period of several windows and ticket volume returns to its normal levels.

4.3 Control Plane & Policy Orchestration

Fast detection alone is meaningless unless followed up by action that is quick, safe, and successful. The analytics tier sends up indicated events to a policy engine that can emit ACL entries, policers, or BGP Flowspec routes, in a DANZ-centric approach. Flowspec enables the distribution of match-action rules based on routing semantics. Match-action points allow rate limits, traffic redirects, or even discard actions to be distributed across the fabric, and remote-triggered black holes can define a final destination of last resort to sink flooding through destination prefixes threatening the stability of uplinks. In production environments, operators set up role-based guardrails that restrict access to who can push global drops. Flowspec withdrawals and RTBH clears require dual authorization.

The difference between orchestration that is tetration-centric is in the locus of control. The policy graph is in the form of allowed communications at the application granularity, which is translated to endpoint controls enforced by agents, virtual switches, or fabric integration points. Staged rollout, simulation, and change windows minimize the possibility of false positives and the disruption to business-critical services that would typically result [35]. In terms of latency, the following occur: anomaly, policy decision, agent update, kernel table change, and first packet drop. Since this route bypasses switch rule churn, the reduction in enforcement latency will skew towards the device update cycle and the scheduling of the agents, which can be planned and budgeted in advance when deploying the canary rollout.

Whichever the paradigm, teams develop a formal control-loop budget. The 150-millisecond error bar on close of the detection window, 50 milliseconds on controller rendering, 150 milliseconds on device or agent update, and 50 milliseconds on verification of anomaly yields a 400-millisecond goal on anomaly to effective mitigation. Budgets are continuously measured; the controller exports per-stage timings, retry counts, and error taxonomies to enable service-level objectives to be enforced. Dependencies on updates are executed sequentially to avoid states of mixed orders, and controllers log causal links in order to track down failed sequences post-mortems. Horizontal context: keeping the short-term context of flows comes up analogous to dynamic memory methods to boost inferences over episodes [26].

4.4 Mitigation Techniques & Playbooks

Gamekeepers can still drop the fastest protective action in the form of ACL drops. In case of coarse volumetric floods, rules matching destination addresses, UDP payload sizes, or reflection vectors are pushed at the edges of the fabric. CP-based traffic control protects routing and telemetry protocols by traffic-policing protocol traffic like BGP, LLDP, and SNMP, but thresholds have to be set to avoid starving keepalive. As attackers use floods to target AMPs, BGP Flowspec offers precise remediation with a clean withdrawal of the highest attack levels, and redirection of suspicious tuples to scrubbing infrastructure when possible, and a policy change to restore baseline policy once traffic has settled.

As shown in the figure below, control-plane monitoring and analytics are used to provision a controller, which programs flow tables with fast ACL drops to match coarse volumetric floods, destination prefixes, UDP payload sizes, and common reflection vectors on fabric edges. Control-plane policing protects BGP, LLDP, and SNMP via carefully selected thresholds that will not turn off keepalives. BGP Flowspec offers accurate, revocable remediation and can redirect suspect tuples to an external scrubbing fabric. Policies are automatically rolled back to baseline after the attack is abated to avert continued disruption to normal forwarding behavior and stability.

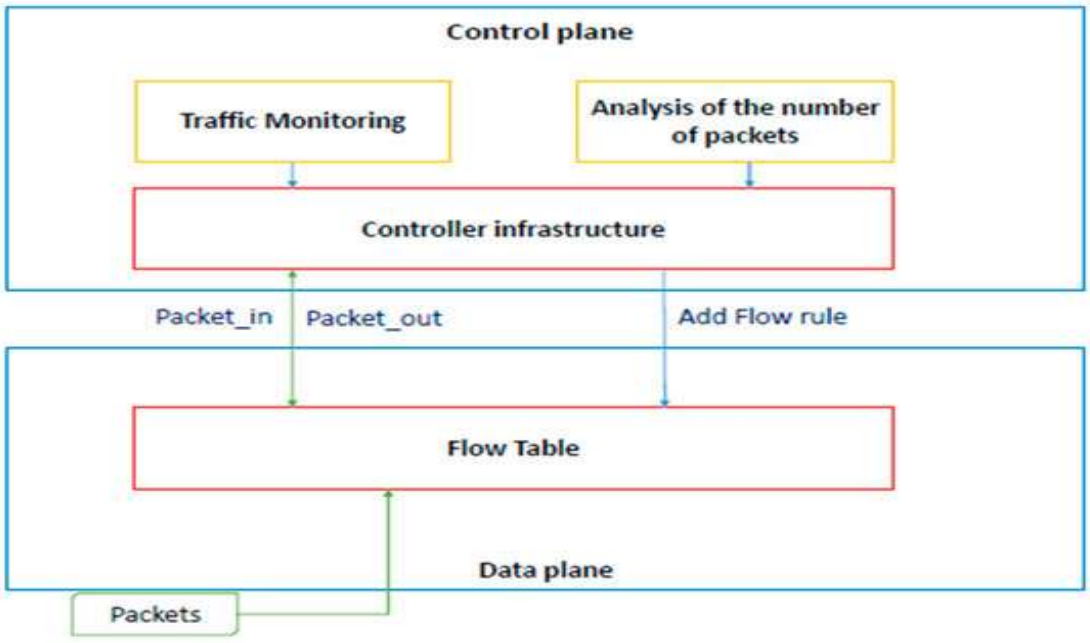

Figure 4: Control-plane monitoring and flow-rule pushes for ACL/Flowspec mitigation

An additional technique that supplements binary drops is the use of rate-limiters when traffic contains both legitimate and malicious traffic, maintaining the same tuple. At the first hop, token buckets flatten burst peaks, giving the higher layers more time to discriminate. At the application layer, rate-limiters and circuit breakers within API gateways or service meshes conserve the capability of the upstream. Where TLS is disallowed from deep inspection, per-certificate or Server Name Indication-based limits offer workable heuristics; QUIC uses connection-ID scoped controls to improve stickiness and fairness. Geographic concentration is absorbed and distributed through content delivery networks, and peak capacity is diluted without revealing the capacity internal to the origin.

Guardrails and learning are encoded in the operational playbooks. All actuators self-simulate where possible; false-positive spikes identified by synthetic transactions and golden dashboards result in automatic rollback; and rate-of-change thresholds apply to policy updates to avoid oscillation. Much of the datasets used to tune detectors need frequent updating, and at a significant cost. Pretraining on unlabeled telemetry with self-supervision may be used to model regular fabric structure and can enhance the ability of anomaly discriminators prior to labeling [18]. To minimize blast radius, feature flags, rollbacks, models, and thresholds are versioned and promoted through the environments. These methods minimize the training debt and speed adaptation to the changing threat landscapes [33].

5. Experiments and Results

5.1 Testbed Setup

The implementation was performed on a small scale, production-like leaf-spine fabric-2 spine and four leaf switches in a non-blocking CLOS with 100 GbE uplinks and 25 GbE downlinks as highlighted in Table 2. Twelve dual-homed hosts were connected: eight to traffic generation/sinks and four to telemetry collection/storage and policy controllers. All hosts were equipped with hardware-timestamping network interface cards, all linked to a PTP Grandmaster that had boundary clocks on all switches. This became a discipline whose interventions produced sub-microsecond consensus on the direction of delay and generated no ambiguity due to asymmetries in paths. During the network stress, links were run at line rate with ECMP enabled, DCB off, and with ECN marking enabled so that the behavior of the queues under stress could be observed. NIC offloads are consistent across all the runs; GRO/LRO turned off on capture interfaces to retain per-packet synchronicity.

Table 2: Leaf–spine testbed for low-latency DDoS evaluation

Area	Implementation	Purpose	Key Parameters
Fabric topology & links	2 spines / 4 leaves, non-blocking CLOS	Production-like data center	100 GbE uplinks, 25 GbE downlinks
Hosts & roles	12 dual-homed servers	Traffic + control separation	8 gen/sinks; 4 telemetry/controllers

Area	Implementation	Purpose	Key Parameters
Timing & capture fidelity	PTP Grandmaster with boundary clocks; HW-timestamp NICs; GRO/LRO off (capture)	Sub-µs timing, per-packet synchronicity	Single PTP domain across all nodes
Forwarding/queuing setup	ECMP on; DCB off; ECN marking on; line-rate tests	Observe queue behavior under stress	Consistent NIC offloads across runs
Traffic & attacks	TRex + MoonGen (Poisson, microbursts)	Precise arrivals and stress variety	UDP floods to 80 Gbps; SYN 20 Mpps; HTTP/1.1 slowloris; HTTP/2 rapid-reset; background 30–40 Gbps
Telemetry, storage & control	Switch flows + host sensors → 3-node Kafka → columnar TSDB; dual passive taps for ground truth; OOB management	Durable analytics, validated measurements, resilient control	Real-time dashboards (cardinality, queues, p50/p95/p99.9, E2E/OWD); Arista DANZ + Cisco Tetration integrations

Traffic generation for stateful TCP/UDP profiles. Traffic generation employed TRex and Poisson arrivals with MoonGen to achieve precise arrivals and controlled microburst generation. The four attack families were run on independent trials and were: stateless UDP floods that flooded to 80 Gbps; randomized-source SYN floods at 20 Mpps; HTTP/1.1 slowloris with partial HTTP headers and stalled sockets; and HTTP/2 rapid-reset sequencing, putting more pressure on the CPU and memory rather than the link. Background east-west traffic was emulated using databases, RPC, message queues, and streaming with 30-40 Gbps aggregate and diurnal-style variability to tempt detectors to differentiate between loads and attacks. A pair of passive taps was used to reflect the ingress and egress of the victim rack to a lossless capture appliance as the ground truth [30].

Telemetry and storage were decoupled with analytics to avoid blocking the head of the line. Switch flows and host sensor data were written to a queryable three-node Kafka cluster and indexed in a columnar time-series data store. Flow cardinality, queue occupancy, p50/p95/p99.9, and end-to-end and one-way latency were rendered on real-time dashboards. Vendor integrations included an Arista DANZ Monitoring Fabric collector/controller to enforce the fabric, and a Cisco Tetration analytics cluster comprising host sensors on the application servers. APIs related to management and policy were segregated on an out-of-band network to ensure that control traffic would be insulated against attack traffic overload.

5.2 Metrics & Measurement Methodology

The first is mitigation and was measured as a closed control loop - sense, decide, and act. Detection time is the time difference between the first telemetry window that surpassed a set anomaly threshold and the machine-readable alert sent over the automation bus [23]. There may not be any relationship between enforcement time and the first confirmed drop/redirect on the victim ingress, so both mirrored packet traces and device counter deltas are used as a verification of any first drop/redirect detection. Time-to-mitigation was identical and reported at p50 and p95. A baseline performance was measured for each workload in the absence of attacks to obtain per-service rate, entropy, and latency envelopes.

Fabric flows had customary intercepts of 200 ms and host sensors of 100 ms, except where otherwise specified. Entropy, burst, and divergence detectors were driven by analytical windows of 250 ms with 50% overlap. Acceptable five-tuple entries were alternated with prefix-wide blocks to induce Rule-churn stress and exercise all TCAM allocation, compaction, and eviction paths. The rate-limited deduplicated notifications to users to avoid alert storms during microbursts, which are principles of scheduling alert bundles that indicate that alerts that are timed and grouped improve downstream actionability and reduce fatigue [28].

All time taking was done using hardware timestamps exposed to user space through bypass, thereby eliminating instrumentation bias in time taking. Pre- and post-mitigation packets were combined by five-tuple, and in the case of TCP, sequence/acknowledgment correlation. Every scenario included a 60-second baseline60-second baseline, an 180-second attack plateau, and a 120-second recovery period, which were replicated ten times with fixed seeds to calculate confidence intervals. Noise characterization was performed by re-running benign-only conditions and injecting sub-threshold bursts to ensure that the detectors were stable. The collectors and controller hosts were pinned to performance CPU governors; the IRQ affordability and RSS were pinned to eliminate measurement skew due to scheduling jitter.

5.3 Comparative Results: DANZ vs. Tetration

Fabric first instrumentation was popular with Volumetric UDP floods. Using DANZ with the sampling rate of 1:1024 and export time of 200 ms, the median detection time was 420 ms (p95 710 ms). An improved sampling of 1:2048 increased median detection to 760 ms at the expense of collector CPU and exporter bandwidth. ACLs were then synthesized and published to the controller, with a median enforcement to first drop 180 ms (p95 310 ms), resulting in a time-to-mitigation of about 600 ms on aggressive configuration and 940 ms at 1:2048. The packet loss during the first second of the attack decreased by 61% as compared to when there was no mitigation.

In the same UDP floods, Cisco Tetration host sensors took more time to detect increasing per-socket receive queues and request-rate abnormalities. With the short flow-record interval settings, median detection was 880 ms (p95 1.3 s). Enforcement time, which consists mainly of rule validation and dependency checks over the host array, was 420 ms median (p95 730 ms). As a result, time-to-mitigation was close to 1.3 s median, with residual packet loss recuperating more slowly as enforcement was at the hosts, rather than in the fabric. The advantage was an increase in precision: other benign neighbors in the same leaf never witnessed collateral drops.

The fund experiments in the form of a SYN flood reduced the difference. DANZ entropy and SYN/ACK divergence tripped at 380 ms median with 1:1024 sampling, whereas host sensors based on SYN backlog pressure and retransmit ratio tripped at 470 ms median. The fastest-propagating enforcement was in the fabric, where notification of the first drops was available 210 ms after invoking the API. The total time of Host enforcement was accomplished in 430 ms in the median [15]. Both systems had low False favorable rates at service- and diurnal-aware thresholds; only one false alert was produced, and it succeeded in rolling back within seconds.

SYN flood is a form of flooding where the bulk of packets is of the SYN type that floods the Maximum Server Handshake of the destination system, and thus depletes the resources and does not allow genuine customers to start a connection, as shown in the figure below. DANZ entropy and SYN/ACK divergence indicated at a 380-ms median with 1:1024 sampling, whereas host sensors based on backlog pressure and retransmit ratio indicated at a 470-ms median. Both systems displayed very low false alerts, in addition to a rollback in low seconds, fabric enforcement achieved around 210-ms delay in propagating first drops, and the host-side enforcement completed at around 430-ms median.

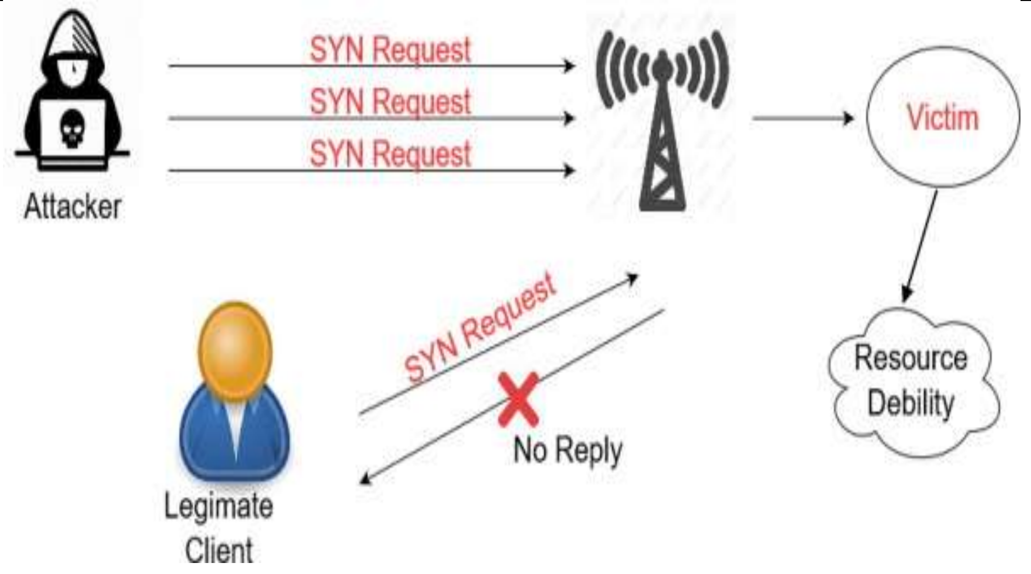

Figure 5: SYN flood attack starving resources, blocking legitimate client connections

In the application-layer cases, the ranking was switched. To 120,000 concurrent partial connections, Tetration demonstrated a 520 ms median detection and 610 ms median enforcement as slowloris exhausted file descriptors and slowed p99.9 latency to well below target. DANZ detection lagged at 1.1 s median at 1:1024 sampling since there were so few requests that entropy changes were obscured; more aggressive 1:512 sampling reduced this value to 670 ms median, but at elevated exporter and collector load. In the case of HTTP/2 rapid-reset, host-level signatures were seen to trigger earlier than fabric-level rate/entropy signatures, and the more locally specific host-rules were less likely to penalize a bulk transfer flow with the same prefix.

When background microbursts were simultaneously executed with a low-and-slow attack, the headroom in both controllers and dataplanes became crucial. DANZ reduced the p99.9 one-way-delay head by 52 percent compared to no mitigation, while detection increased 140 ms during some of the most enormous bursts as collectors rebalanced partitions. Tetration implemented a staged deployment of non-critical rules based on controller load, maximizing host CPU by adding 180 ms to the eventual completeness of enforcement across each of the target groups. The trade-offs implied by the contrast reflect other optimization tasks, whereby the dispensing of batching and sorting of routes enhances the throughput at the expense of jamming latencies during rush hours [21].

The use of Resources was monitored on an inclusive basis. With the most active TCAM, transient TCAM utilization was highest (64 percent) when installing rules for both prefix and five-tuple, including expanding the TCAM by longest-prefix match and consolidating port ranges, which cut peak utilization by a third with no detectable accuracy loss. The collector CPU reached a high of 78% at the most aggressive sampling. At Tetration, the host-agent CPU stayed below 3%, and it spiked to 7% at times when connection tracking came in one burst. Controller correlated policy-compile time with rule-set cardinality; the performance benefit of 50-100 rules per commit batch p95 enforcement gain by approximately 22% without correctness.

5.4 Sensitivity, Scale & Failure Testing

Sensitivity variation included varying ratios of sampled over export rates, export times, and window sizes on the detector. On DANZ at 1:512 and 100 ms export, median detection was down 34 percent relative to default [36]. However, exporter bandwidth was up 0.6 percent of line rate per monitored interface, and collector CPU was up 28 percent. Even at 1:2048 and an export time of 300 ms, the detection doubled in the case of low-rate application attacks, but volumetric attacks remained less than one second. Detector windows that were less than 150 ms were noise-sensitive, whereas windows longer than 400 ms produced delayed onset recognition but with no significant decline in false positives.

Scale tests were subjected to policy systems and pipelines. The fabric controller could handle approximately 5,000 active rules with a sub-400 ms median installation time before exceeding a p95 of over one second, as the previously mentioned TCAM remapping paths were utilised. The throughput of the analytics pipeline was 2.5 million flow updates per second until back-pressure alerted exporters to enlarge sampling. The Tetration cluster applied more than 20,000 micro-segmentation rules with steady medians; with bursty change sets, p95 enforcement took longer than a second unless commit batching was active, in which case, p95 fell to less than 800 ms.

Both PTP degradation and collector lag were used as failure tests, and a host-agent fault was injected. PTP degraded into free-running oscillators, which caused a drift in one-way delay, and a broadening of cross-correlation between mirrored capture points by 80-120 ms, slightly increasing detection times and obscuring microbursts. By slowing down Kafka I/O, emulated collector lag, with the one-second lag enough to push DANZ detection back approximately to its original time, but watchdogs triggered loss of signal, moving detectors to coarser features. At a lapse of three tracks, each system issued health warnings and stopped churning non-essential rules. Attacks via the Agent crashing a subset of hosts made Tetration less visible to lateral and low-and-slow traffic; nevertheless, fabric-level protection was able to contain volumetric flooding. With failures in all planes, policy traffic remained stable with control-plane isolation, and the suspect rules were returned within seconds based upon counter deltas and telemetry revived.

6. Discussion

6.1 Interpretation of Results

Comparative analysis shows that a fabric-first system that uses Arista DANZ construction reduces the sense-decide-act cycle time during high-rate layer-3/4 floods. Switch ASICs export counters and sampled flows at very fine granularities over sFlow/IPFIX and gNMI, collectors consume these streams without the need to context switch between hosts, and mitigation policies are compiled to run in TCAM or BGP Flowspec with little serialization. Effectively, volumetric UDP amplification and SYN floods are immediately recognized as fan-out, heavy-hitter, or entropy-drop patterns; enforcement then becomes a matter of in-fabric drop, policer, or diversion actions before the congestion inflow generally reaches a critical point. False negative sensitivity to packet processing back-pressure on servers is lower because the decision is based on deterministic queue and flow statistics instead of host availability, and mitigation does not require per-workload responsiveness or agent health.

These data also show that a host-offense-centric method, exemplified by Cisco Tetration agent model and policy graph, is expected to be superior in both application-layer attacks of the low-and-slow variety and during lateral movement. Devices can provide kernel-level telemetry and process context, which reveal request pacing, connection churn, and dependency on followers of other services that are not visible through pure flow statistics. Micro-segmentation rules may restrict east-west access very specifically within the range of the threatened workload, to avoid credential-stuffing, inventory scraping, or sneaky enumeration below fabric sampling thresholds. In insider or post-compromise situations, host provenance and software identity aid in attribution and allow finer-grained least-privilege containment in the host than would be possible in the switching fabric. As a result, Tetration is preferable in scenarios where semantics leads to decisions, whereas DANZ is preferable in cases of speed and a large spectrum of blast-radius control.

A combination pocket, then, is the majority of the path of the hybrid trigger. Fabric telemetry must act as the initial guard against sudden volumetric swings and automatically put coarse rate limits, ACL drops, or diversion pathways in place and alert host sensors to a narrower scope at the process, user, and service level. Fabric caps should, on the other hand, be sought in the case of host anomalies when the blast radius tends to fan out. To increase the reliability of thresholds and classifiers, it is helpful to regularly augment them with synthetic but also realistic traces representing rare onsets and corner cases; it has been demonstrated across domains that generated datasets can be used to strengthen rare-event detection without a significant increase in false-positive rates [32]. In general, these results are consistent with the predictions: fabric-first beats during fast floods when there is little room in the latency budget, host-centric beats where application semantics are critical, and a coordinated, bi-directional loop yields the least operational risk.

6.2 Operational Considerations

Operational discipline is what separates hypothetical latency budgets on paper from those attained in production. Policy pushes via staged pipelines should be gate-controlled by policy; to show what would have blocked flows in shadow evaluation in monitor-only mode; to validate TCAM and agent limits in dry-run compilation; to prevent rule storms in bounded commit queues; and to do canary enforcement on a small subset of leafs or service instances before general rollout of a policy. Detector authorship must be separated from enforcement approval, and emergency break-glass privileges must be time-scoped and logged. Golden rule sets - pre-approved ACL/Flowspec templates and micro-segmentation guardrails, minimize the time expended in discussions during incidents, and limit the likelihood of over-fitting to a short-term aberration.

During the accumulation of classification confidence, Brown-out controls are essential. Instead of instantly dropping, hierarchical policers and token buckets may trim off the peak of an attack over a bounded time, buying extra time during which further evidence may be collected to establish availability. Executability SLOs should be clear on each iteration of the loop: freshness of telemetry, detector turnaround rate, policy compilation period, propagation and acknowledgment, and rule programming at switches and at hosts [8]. To implement SLOs, SNOOP provides critical instrumentation: hardware timestamps on mirrored flows, signed controller acks, and counters per device that confirm that rules are active and matching. It also helps to eliminate silent drifts as export periods drift or API delays back-log compile time.

Forensics and Audit must be impossible to change. All changes require the release of a signed artifact that documents alert characteristics, decision reasoning, reviewer information, and the precise diff that affected infrastructure. Kill chain and collateral effects can be reconstructed post-incident by PCAP rings at choke points, synchronised with flow logs and application traces using PTP. Unbooks should specify rollback requirements, rule aging, and garbage collection, as well as a set of standard counters that reveal the unintended impact, such as increases in retransmissions, cache miss spikes, or 429/503 rates.

6.3 Limitations

The test was done in a controlled laboratory, and it may not encapsulate some heterogeneity in production. The workloads running in data-centers tend to have a diurnal aspect, suffer from noisy-neighbor syndrome, and overlapping maintenance windows, which make it hard to attribute them. The data may over-sample traditional floods and under-sample emerging vectors like HTTP/2 rapid-reset and HTTP/3/QUIC overloads; flow-level timing and size observability can be limited by encryption, which impairs detectors based on semantics at the header level. Additionally, mirroring and ERSPAN employed to confirm measurements can interfere with queues differently than production inline paths. Absolute timing may be skewed due to vendor default, pipeline depth, or NIC offloads in the testbed [19]. These limits imply a warning when drawing specific latencies, but qualitative conclusions of where the fabric-first and host-centric models seem to win remain sound.

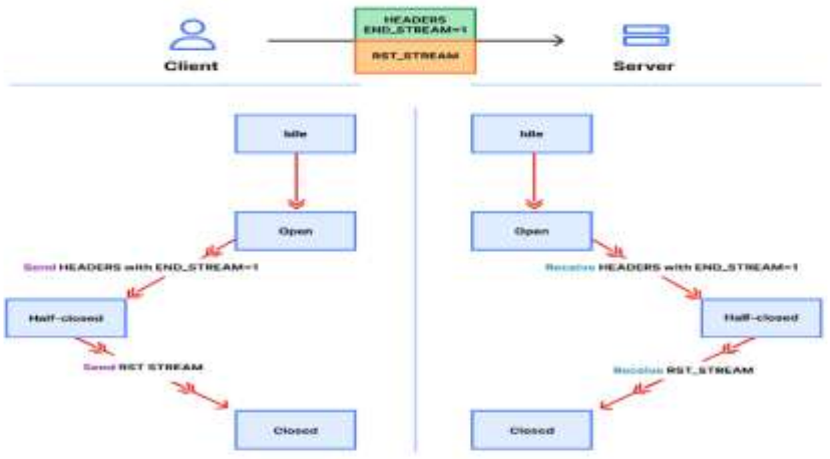

Figure 6: HTTP/2 rapid-reset lifecycle—an under-sampled emerging DDoS vector

As shown in the figure above, the HTTP/2 rapid-reset is an example of an emergent attack surface that the study under-sampled: clients open streams, then set END_STREAM=1 and send RST_STREAM, which requires state churn on the server side. Since the assessment was done in a laboratory, such behaviors may be different when diurnal loads, noisy neighbors, and maintenance overlaps are considered during production [5]. The header-level observability can also be further reduced by encryption, and the queues can be perturbed by ERSPAN/mirroring. Vendor defaults, pipeline depth, and NIC offloads can distort absolute latencies. Still, the overall relative conclusions regarding the efficacy of a fabric-first hosting approach as compared with a host-centric approach will generally hold.

6.4 Future Considerations

Host datapaths with programmable forwarding can enable the loop to be compressed, with higher precision. P4-capable switches can build line-rate and heavy-hitter sketches in network ingresses, and detect anomalous flows and heavy hitters on a per-destination basis without exporting any large samples. In-band network telemetry could add hop-level queue occupancy and timestamps to packets, tools to infer when path inflation has occurred, and where the relevant target enforcement is to be proximate to the packet. Flow-aware Flowspec, policer templates, and per-port admission controls can be pre-staged to create predictable mitigation budgets in environments with high rule churn. eBPF/XDP programs at the host may enforce rate limits and lightweight anomaly filters before packets climb the stack, and user-space suites inject semantic labels used to further micro-segment. Thresholds assisted by machine-learning should be informative and not definitive, with concept-drift monitors that re-establish when workload mixes vary. Synthetic attack traces should be introduced on game days to test detectors and recalibrate alerting, as well as confirm rollback logic under load.

Intra-Cloud scrubbing should have top integration. BGP Flowspec can direct only offending prefixes into provider scrubbers through GRE or VXLAN tunnels, leaving clean traffic with low latency. In contrast, authentic telemetry of the scrubbing service can be fed back, containing disposition and byte counts to enable accurate billing and post-mortems. One policy can define detection confidence, maximum blast radius, and rollback timing as enforcement choices limited to a service-level objective that is intent-based [4]. A unified controller that reasons over joint fabric and host intents, and verifies rules for shadowing and TCAM saturation, as well as simulating changes in a digital twin, will significantly reduce the time taken to mitigate as well as the likelihood of self-inflicted loss.

7. Conclusions

The study shows that high-touch DDoS mitigation effectiveness depends on squeezing the sense–decide–act loop while maintaining accuracy and headroom across data planes and control planes. Volumetric UDP/SYN UDP/SYN Volumetric cases with the fabric-first path via Arista DANZ under sub-second time-to-mitigation, sFlow/IPFIX export time, and ACL/BGP Flowspec programmability. Using 1:1024 sampling and 200 ms exports, median detection was 420 ms (p95 710 ms) and median enforcement-to-first-drop was 180 ms (p95 310 ms), reducing first-second loss by 61 percent. A coarser 1:2048 resized sampling lifted median detection to 760 ms and pushed end-to-end mitigation to ~940 ms. In contrast, Cisco Tetration was better on application-layer low-and-slow and lateral movement attack profiles: host sensors reported 520 ms median detection and 610 ms median enforcement in slowloris studies, whereas fabric signals failed until reasons forcing more aggressive sampling were implemented. SYN flood was the only one where there was convergence (380 ms vs. 470 ms median detection by DANZ and Tetration, respectively), and where both showed low false positives with staged thresholds. Resource tradeoffs were evident: fabric-side TCAM volume peaked at 64 percent but declined by a third using the rule consolidation feature; collector CPU was up to 78 percent under aggressive sampling; Tetration agents consistently averaged under 3 percent CPU, sometimes up to 7. The hybrid operation, with fabric as rough, direct, temporary blast-radius cap and hosts as fine-grained scope, permanently reduced tail latency (e.g., 52 percent p99.9 OWD in microburst situations with DANZ) without generating undue collateral.

Operators to use a hybrid playbook with clear latency budgets per stage and pre-authorized actions. With flood attacks with short ramp-ups, focus first on using fabric enforcement embodying coarse ACLs, policers, and Flowspec redirects to achieve a resolution metric of ≤400 ms anomaly-to-effective-mitigation (e.g., ~150 ms detection-window closure, ~50 ms render, ~150 ms program, and ~50 ms verification). In case of application-layer misuse and data center-to-data movement, rely on host micro-segmentation and identity-based rule sets to constrain the affected services. Retain adaptive sampling (e.g., escalation to 1:512 sampling on triggers) until a given backpressure-aware collector; default to normal afterwards when stability has returned. Prestage golden rule templates (coarse denies, RTBH/Flowspec patterns, token-bucket rate limiters) and impose change discipline: shadow evaluation, dry-run TCAM checks, commit batching (50100 rules per batch improved p95 enforcement by ~22 percent), and canary rollouts. Instrument observability SLOs on telemetry freshness, detector runtime, compile/propagation/programming latencies, and rule-hit confirmations through hardware timestamping and authentically signed controller acknowledgment. Keep the control-plane paths isolated and PTP-synchronized; mirror/ERSPAN throttled to avoid tail latency. Lastly, formalize rules around rollback conditions, rule aging, and garbage collection to prevent back and forth and TCAM starvation as a result of incident churn.

Understanding which of these loci of control to use when becomes easier as the comparison implies that a fabric-first locus of control is best suited to cooked recipes where speed and blast-radius mitigation are desirable under volumetric pressure, a host-first locus of control in encrypted eastwest and L7 patterns where semantics precision matters, and a synergistic loop is ideal in mixed realities. The empirical framework standardized telemetry windows, as well as latency measures of enforced and line-rate stamp timestamps, and repeatable seed replaying, provides a vendor-independent approach to defining and ensuring mitigation SLOs before incidents. Prospectively, the programmable pipelines (P4/INT) can be used to offload the heavy-hitter detection and queue telemetry at ingress. In contrast, the eBPF/XDP can provide per-host rate caps and granular attribution. Policy simulation (through intent-driven controllers that emulate the effects of policy changes), shadowing/TCAM limit checks, and cloud scrubbing orchestration on a per-prefix basis should minimize mitigation latency and self-inflicted loss. Action necessary in practice is game-day validation of synthetic traces, to keep detectors, budgets, and rollback up to date with dynamic traffic and attack trends.

References

[1] Aijaz, A., & Stanoev, A. (2021). Closing the loop: A high-performance connectivity solution for realizing wireless closed-loop control in industrial IoT applications. IEEE Internet of Things Journal, 8(15), 11860-11876.

[2] Arora, A., Pandey, M., Siddiqui, M. A., Hong, H., & Mishra, V. N. (2021). Spatial flood susceptibility prediction in Middle Ganga Plain: comparison of frequency ratio and Shannon's entropy models. Geocarto International, 36(18), 2085-2116.

[3] Chavan, A. (2023). Managing scalability and cost in microservices architecture: Balancing infinite scalability with financial constraints. Journal of Artificial Intelligence & Cloud Computing, 2, E264. http://doi.org/10.47363/JAICC/2023(2)E264

[4] Chowdhary, A. (2020). Software-defined Situation-aware Cloud Security (Doctoral dissertation, Arizona State University).

[5] Clancy, B. M., Theriault, B. R., Turcios, R., Langan, G. P., & Luchins, K. R. (2023). The effect of noise, vibration, and light disturbances from daily health checks on breeding performance, nest building, and corticosterone in mice. Journal of the American Association for Laboratory Animal Science, 62(4), 291-302.

[6] *Dyer, M. J. (2020). A telescope control and scheduling system for the Gravitational-wave Optical Transient Observer. arXiv preprint arXiv:2003.06317.*

[7] *Finstad, R. (2020). Implementation of network moving target defense in embedded systems (Master's thesis, Iowa State University).*

[8] *Ghit, R. (2021). Monitoring serverless applications: an SLO-based approach (Doctoral dissertation, University of Stuttgart).*

[9] *Huang, W., Yin, K., Ghorbanzadeh, M., Ozguven, E., Xu, S., & Vijayan, L. (2021). Integrating storm surge modeling with traffic data analysis to evaluate the effectiveness of hurricane evacuation. Frontiers of Structural and Civil Engineering, 15(6), 1301-1316.*

[10] *Kappal, S. (2019). Data normalization using median median absolute deviation MMAD based Z-score for robust predictions vs. min–max normalization. London Journal of Research in Science: Natural and Formal, 19(4), 39-44.*

[11] *Karwa, K. (2023). AI-powered career coaching: Evaluating feedback tools for design students. Indian Journal of Economics & Business. https://www.ashwinanokha.com/ijeb-v22-4-2023.php*

[12] *Keskinen, S. (2022). Cloud services utilization in Pension Insurance business.*

[13] *Kim, T., Kwon, T., Lee, J. U. N., & Song, J. (2021). F/wvis: Hierarchical visual approach for effective optimization of firewall policy. IEEE Access, 9, 105989-106004.*

[14] *Konneru, N. M. K. (2021). Integrating security into CI/CD pipelines: A DevSecOps approach with SAST, DAST, and SCA tools. International Journal of Science and Research Archive. Retrieved from https://ijsra.net/content/role-notification-scheduling-improving-patient*

[15] *Kosek, M., Doan, T. V., Granderath, M., & Bajpai, V. (2022, March). One to rule them all? A first look at DNS over QUIC. In International Conference on Passive and Active Network Measurement (pp. 537-551). Cham: Springer International Publishing.*

[16] *Kumar, A. (2019). The convergence of predictive analytics in driving business intelligence and enhancing DevOps efficiency. International Journal of Computational Engineering and Management, 6(6), 118-142. Retrieved from https://ijcem.in/wp-content/uploads/THE-CONVERGENCE-OF-PREDICTIVE-ANALYTICS-IN-DRIVING-BUSINESS-INTELLIGENCE-AND-ENHANCING-DEVOPS-EFFICIENCY.pdf*

[17] *Lin, Y. B., Tseng, C. C., & Wang, M. H. (2021). Effects of transport network slicing on 5G applications. Future Internet, 13(3), 69.*

[18] *Liu, D., & Abdelzaher, T. (2023). Self-Supervised Learning from Unlabeled IoT Data. In Artificial Intelligence for Edge Computing (pp. 27-110). Cham: Springer International Publishing.*

[19] *Liu, M., Cui, T., Schuh, H., Krishnamurthy, A., Peter, S., & Gupta, K. (2019).*

Offloading distributed applications onto smartnics using ipipe. In Proceedings of the ACM Special Interest Group on Data Communication (pp. 318-333).

[20] Ma, Y., Gu, M., Chen, L., Shen, H., Pan, Y., Pang, Y., ... & Sun, L. (2021). Recent advances in critical nodes of embryo engineering technology. Theranostics, 11(15), 7391.

[21] Nyati, S. (2018). Revolutionizing LTL carrier operations: A comprehensive analysis of an algorithm-driven pickup and delivery dispatching solution. International Journal of Science and Research (IJSR), 7(2), 1659-1666. Retrieved from https://www.ijsr.net/getabstract.php?paperid=SR24203183637

[22] Nyati, S. (2018). Transforming telematics in fleet management: Innovations in asset tracking, efficiency, and communication. International Journal of Science and Research (IJSR), 7(10), 1804-1810. Retrieved from https://www.ijsr.net/getabstract.php?paperid=SR24203184230

[23] Owens, D., Abeysirigunawardena, D., Biffard, B., Chen, Y., Conley, P., Jenkyns, R., ... & Thorne, M. (2022). The oceans 2.0/3.0 data management and archival system. Frontiers in Marine Science, 9, 806452.

[24] Paparrizos, J., Liu, C., Barbarioli, B., Hwang, J., Edian, I., Elmore, A. J., ... & Krishnan, S. (2021, January). VergeDB: A Database for IoT Analytics on Edge Devices. In CIDR.

[25] Raheem, M. (2019). Mitigation of inter-domain Policy Violations at Internet eXchange Points.

[26] Raju, R. K. (2017). Dynamic memory inference network for natural language inference. International Journal of Science and Research (IJSR), 6(2). https://www.ijsr.net/archive/v6i2/SR24926091431.pdf

[27] Rios, V. D. M., Inácio, P. R., Magoni, D., & Freire, M. M. (2022). Detection and mitigation of low-rate denial-of-service attacks: A survey. IEEE Access, 10, 76648-76668.

[28] Sardana, J. (2022). Scalable systems for healthcare communication: A design perspective. International Journal of Science and Research Archive. https://doi.org/10.30574/ijsra.2022.7.2.0253

[29] Sardana, J. (2022). The role of notification scheduling in improving patient outcomes. International Journal of Science and Research Archive. Retrieved from https://ijsra.net/content/role-notification-scheduling-improving-patient

[30] Sharp, R. (2023). Network Security. In Introduction to Cybersecurity: A Multidisciplinary Challenge (pp. 171-233). Cham: Springer Nature Switzerland.

[31] Shi, X., Liu, W., He, L., Jin, H., Li, M., & Chen, Y. (2020). Optimizing the SSD burst buffer by traffic detection. ACM Transactions on Architecture and Code Optimization (TACO), 17(1), 1-26.

[32] Singh, V. (2021). Generative AI in medical diagnostics: Utilizing generative models to create synthetic medical data for training diagnostic algorithms. International

Journal of Computer Engineering and Medical Technologies. https://ijcem.in/wp-content/uploads/GENERATIVE-AI-IN-MEDICAL-DIAGNOSTICS-UTILIZING-GENERATIVE-MODELS-TO-CREATE-SYNTHETIC-MEDICAL-DATA-FOR-TRAINING-DIAGNOSTIC-ALGORITHMS.pdf

[33] Singh, V. (2023). Enhancing object detection with self-supervised learning: Improving object detection algorithms using unlabeled data through self-supervised techniques. International Journal of Advanced Engineering and Technology. https://romanpub.com/resources/Vol%205%20%2C%20No%201%20-%2023.pdf

[34] Souza, P. S. S. D. (2023). Minimizing latency and maintenance time during server updates on edge computing infrastructures.

[35] Thompson, M. J., Bennett, A. R., Williams, D. K., Carter, E. A., & Martin, S. (2020). Role of Machine Learning in Predicting Downtime During Data Migration.

[36] Vanzomeren, C. M., Acevedo-Mackey, D., Murray, E. O., & Estes, T. J. (2019). Maintaining Salt Marshes in the Face of Sea Level Rise-Review of Literature and Techniques.

[37] Wan, Y., Song, H., Xu, Y., Zhang, C., Wang, Y., & Liu, B. (2021, May). Adaptive batch update in TCAM: How collective optimization beats individual ones. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications (pp. 1-10). IEEE.