Moving Target Defense for DDoS Mitigation with Shuffling of Critical Edge(s) Connections

Amir Javadpour, Forough Ja'fari, Tarik Taleb, and Chafika Benzaïd

***Abstract*—Moving Target Defense (MTD) has as a widely adopted approach to mitigate vulnerability exploitation. It is a widely adopted approach to mitigate the exploitation of vulnerabilities. Its dynamic and proactive nature makes it well-suited for SDNs requiring comprehensive and continuous monitoring. A core objective of MTD is to minimize the number of hosts shuffled while maintaining robust security and low scrambling frequency. This paper introduces a novel approach, the Number of Edge Connections (NoEC) strategy, aimed at mitigating Distributed Denial of Service (DDoS) attacks in a resource-efficient manner. This is achieved by strategically reconfiguring a select group of highly connected hosts known as "Edges" to protect critical assets. This approach enhances analytical clarity and supports informed selection of defense strategies tailored to specific edge deployment scenarios. We designed a system utilizing NoEC and conducted simulations using Mininet. The results show that NoEC reduces the complexity by 55.12% compared to previous MTD methods while increasing the security level by 15.72%. Among the techniques, topology randomization and edge node shuffling show the highest disruption effect, validating the approach's practical viability and robustness in defending edge infrastructures.**

***Index Terms*—Moving Target Defense, DDoS Mitigation, Shuffling Edge, Mininet, Software-defined networks**

I. Introduction

Software-defined networking (SDN) is reshaping computer networking by improving network services such as management, monitoring, virtualization, distribution, and integration. This advancement is closely associated with the advent of technologies. In SDN, a centralized unit known as a controller manages network traffic control. It can enforce and establish rules on switches to guide detailed network processes and traffic [1]. Despite these advances, SDNs face several security challenges, particularly DDoS attacks. These incidents represent advanced and damaging types of cyber threats, known for being widespread and distributed attacks. They have become more frequent and are often leveraged for various malevolent purposes [2, 3, 4, 5]. Consequently, developing and implementing security measures to counter DDoS attacks is important. MTD is a proactive strategy aimed at safeguarding critical assets from DDoS attacks by dynamically altering the attack surface. MTD seeks to thwart attackers by changing the attack surface through methods like network address shuffling, rendering any intelligence gathered during network reconnaissance obsolete [6, 7, 8]. Compared to other security mechanisms, MTD offers several advantages: (1) scalability, (2) minimal need for threat detection, and (3) frustration with adversaries. Hence, creating a network capable of altering its structure to apply MTD strategies is complex. However, with the dynamic and manageable framework provided by SDNs, especially with the advent of 5G/6G technologies, it becomes a suitable environment for deploying dynamic security mechanisms such as MTD approaches.

Developing Edge(s) Connections for the DDoS Mitigation method is critical because of the trade-off between defensive benefits and associated costs. The main costs for MTD are driven by the frequency of reconfiguration and the algorithm's complexity. Previous research has focused on complex network features to minimize the number of reconfigurations. The current approaches have mostly failed to simplify the method, and the runtime tends to increase as the network grows. With the advancement of 5G/6G technologies, addressing the cost and complexity challenges for efficient and scalable MTD solutions is pivotal.[9, 10, 11].

In this paper, we present a novel MTD shuffling method utilizing Edge Connections for DDoS Mitigation to reduce costs by targeting and relocating the most cost-effective compromised hosts. This strategy depends on identifying low-cost hosts based on their connections to critical servers, where these connections are represented as "edges" in a model. Our method focuses on key hosts to achieve higher security effectiveness while reducing costs. In SDN networks, the NoEC solution effectively addresses unique challenges such as massive scale, low latency, and dynamic environments. It manages many connected devices, while minimizing costs and complexity. NoEC's prioritizes critical connections, making it ideal for scalable security mechanisms in these networks. Its cost-effectiveness also helps mitigate DDoS attacks more efficiently than traditional methods.

The main contributions are as follows:

1) **MTD Framework with Six Shuffling Techniques:** We propose an edge-oriented MTD framework using six dynamic shuffling strategies to resist DDoS attacks. This integration improves adaptability and system unpredictability.
2) **Graph-Based Attack Propagation Modeling:** A novel graph model illustrates how each MTD technique alters attack paths and exposure. This helps quantify disruption effectiveness in adversarial settings.
3) **Visual and Comparative Analysis of MTD Methods:** We provide visual simulations comparing MTD tech-

Amir Javadpour (Senior Cybersecurity Researcher MOSA!C Lab / ICTFICIAL Oy, Finland). **Forough Ja'fari** (Sharif University of Technology). **Tarik Taleb** (Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum). **Chafika Benzaïd** (Faculty of Information Technology and Electrical Engineering, University of Oulu)

Corresponding author: Amir Javadpour (a.javadpour87@gmail.com)

niques under attack. These help identify the most robust methods for different scenarios.

4) **Lightweight Shuffling Protocols for Edge Systems:** Our protocols ensure minimal latency and overhead, making MTD feasible in constrained edge environments. This enhances practical deployability.
5) **Simulation-Based Evaluation with Realistic Metrics:** We implement a DDoS simulation and evaluate MTD with quantitative metrics like attack spread and shuffling efficiency. Results validate our defense strategy.
6) **Foundation for Adaptive and Optimized MTD Policies:** Our work sets the stage for future learning-based or game-theoretic optimization of MTD decisions. It enables smarter, adaptive cyber defense systems.

Gaps: Despite a growing body of work on moving-target and shuffling-based defenses, operators still lack a practically deployable mechanism that can be plugged into SDN controllers and reason about which edge hosts to reconfigure under realistic 5G/6G traffic. Existing proposals either assume simplified topologies, ignore slice heterogeneity, or focus on address and VM randomisation without providing an explicit selection logic for high-impact edge hosts. This leaves a gap between conceptual MTD strategies and a concrete, connection-driven defence that network operators can use to tune disruption and overhead in a systematic way.

Motivation: Moving-target and reassignment defenses have shown that continuously changing the attack surface can hinder adversaries, but their deployment in SDN-based 5G/6G networks remains challenging. Existing schemes typically operate at the level of IP addresses, proxy pools, or VM migration, which is either too coarse-grained or too costly when thousands of edge hosts and multiple network slices coexist. They offer limited guidance on *which specific edge connections or hosts* should be reconfigured at each epoch to maximally disrupt coordinated attacks while keeping collateral impact on benign traffic low. Furthermore, most current approaches are not explicitly slice-aware and therefore struggle to respect the heterogeneous QoS requirements of URLLC, eMBB, and mMTC services during reconfiguration. These limitations motivate a defence mechanism that reasons directly on edge connections, selects a small set of high-impact hosts to reconfigure at each step, and remains compatible with slice-specific latency and reliability guarantees. NoEC is introduced in this work precisely to meet these requirements.

Operationally, our goal is an MTD policy that (i) runs with near-linear per-epoch overhead, (ii) honors explicit guardrails on rule churn and QoS (cf. Sec. VIII-E), and (iii) exposes simple knobs (budget B, cadence T_{shuf}) suitable for SDN/5G controllers. From a game-theoretic perspective, the defender repeatedly chooses a reconfiguration action under budget while the attacker adapts scanning/targeting. Our cost-normalized rule $\Psi(h) = \deg_S(h)/c(h)$ acts as a *deployable best-response heuristic*: it maximizes structural path coverage per unit cost without solving a full-information game, and it can be embedded as the defender's action in repeated or Stackelberg formulations. Trust/risk signals (Sec. IV-J) serve as priors by shaping the effective cost $c'(h)$, thereby inducing mixed or randomized strategies when jitter is enabled (Sec. VIII-E). This bridges practical MTD (guardrails, reproducibility) with game-theoretic intent (adaptive, budgeted defense) while keeping the controller footprint small.

The paper is structured as follows: Section II reviews MTD techniques and their limitations. Section III details the proposed NoEC method. Section IV discusses implementation strategies for NoEC in an SDN environment. Section V presents simulation results comparing NoEC to other MTD approaches. Finally, Section VI concludes with key findings.

II. Related Work

This section reviews prior research on applying MTD techniques within SDN environments to mitigate cybersecurity threats. A summary of these studies is provided in Table Table I.

Steinberger et al. [12] implemented an MTD approach to demonstrate its effectiveness in reducing the success rate of DDoS attacks within SDN environments. Building upon this, Luo et al. [13] proposed an integrated framework that combines MTD with honeypots to enhance network security against DDoS threats in SDNs. Aydeger et al. [14] introduced an optimized MTD strategy modeled as a signaling game to mitigate DDoS attacks, while Zhou et al. [15] similarly employed game-theoretic modeling, formulating the MTD mechanism as a three-sided game. They developed the Tripartite Cost-Effective Strategy Algorithm (TGCESA) using Markov decision processes to balance cost and effectiveness. In large-scale network scenarios, Narantuya et al. [16] leveraged multiple SDN controllers to improve the scalability, security, and performance of MTD strategies particularly relevant in the context of complex and high-speed 5G networks.

In terms of dynamic response techniques, Liu et al. [17] introduced a port-hopping strategy where switches dynamically alter packet source and destination ports to mislead attackers and prevent DDoS attacks. This method is well-suited for managing the high data rates and dynamic traffic patterns of 5G networks. Similarly, Chowdhary et al. [18] employed an MTD-hopping strategy to defend against multi-stage attacks, aligning well with the evolving threat landscape in 5G environments. Shi et al. [19] proposed a flexible MTD framework with adjustable obfuscation levels, enabling adaptation to varying security requirements. Lastly, Debroy et al. [20] introduced a frequency minimization-based MTD approach that secures SDN applications from DDoS attacks by reducing the frequency of defensive reconfigurations an essential factor in preserving system performance in fast-paced 5G networks.

Hyder and Ismail [21] applied MTD techniques to strengthen both the control and data planes within SDN environments. Their approach involved dynamic port and IP forwarding to counter reconnaissance attacks in the data plane an increasingly vital defense mechanism for high-speed, complex 5G/6G networks. Similarly, Medina-López et al. [22] leveraged MTD to detect malicious nodes in peer-to-peer

TABLE I
SUMMARY OF RELATED WORK ON MTD METHODS IN SDNS.

Reference	Evaluation Metrics	Cost	DDoS Mitigation	Notes on 5G/6G Adaptability
[12]	Attack success rate	✗	✓	Focused on traditional networks; may require adaptation for high-speed 5G/6G environments.
[13]	Delay	✗	✓	Limited in handling high-volume traffic typical of SDNs networks.
[14]	Cost, packet loss	✓	✓	Suitable for scalable networks; performance may vary with the high dynamics of 6G.
TGCESA [15]	Delay, packet loss, CPU load	✓	✓	Designed for existing networks; scalability for SDNs needs further exploration.
[16]	Delay, attack probability	✗	✗	May not address the specific needs of ultra-low latency 5G/6G applications.
[17]	Response time, service rate	✗	✓	Could be adapted for high-speed scenarios, but lacks focus on SDNs specific requirements.
[18]	Threat score, service risk value	✓	✗	The method is less focused on DDoS and may need adjustments for 5G/6G traffic patterns.
[19]	Delay, information disclosure	✗	✗	Primarily designed for static networks; might require significant modifications for 5G/6G.
[20]	Packet loss, attack success rate	✓	✓	Effective in current environments but may need enhancements for the high-speed nature of SDNs.
[21]	Defender's success rate	✓	✗	Focuses on defensive metrics; might be adapted to 5G/6G with additional optimizations.
[22]	Detection probability	✓	✗	Detection-focused; integration into 5G/6G networks may require adjustments for latency and speed.
[23]	Latency, reconnaissance cost	✓	✓	Addresses latency issues but may need refinement to match the ultra-low latency needs of 5G/6G.
[24]	Attack graph generation time	✓	✗	Suitable for traditional networks; may face challenges with the complexity of 6G environments.
BAP [25, 26]	Delay, complexity, success rate	✓	✗	Balances delay and complexity; potential for adaptation to SDNs, though specifics are not covered.
IP Shuffling [27]	Detection time, scanning disruption rate	✗	✓	SDN-based; scalable and lightweight for integration into network slices or IoT layers in 5G/6G.
Server Relocation [28]	Service continuity, migration latency	✓	✗	Beneficial in virtualized cloud-native 5G/6G cores; requires orchestration support.
Topology Randomization [29]	Network entropy, reachability	✓	✗	Supports zero-trust in dynamic 6G environments; works well with intent-based networking.
Route Mutation [30]	Attack repetition rate, rerouting overhead	✓	✓	Secure routing suited for 5G/6G backbones with programmable dataplanes.
Port Hopping [31]	Port scan success rate, latency	✗	✓	Lightweight defense for delay-sensitive SDNs applications; especially effective in edge environments.
NoEC	Complexity, adversary's success rate, rate of compromised servers	✓	✓	Optimized for low complexity and high security; well-research for dynamic SDNs environments with scalability in mind.

SDNs by dynamically modifying the destination IP addresses of exchanged messages, thereby addressing the evolving adaptive security requirements of 6G networks.

To manage high-speed demands in SDNs, Chang et al. [23] proposed a cost-efficient MTD strategy that randomizes IP addresses and employs hash-based signatures to synchronize different MTD phases across the network. In a complementary direction, Chowdhary et al. [24] utilized SDN controller-driven network reconfiguration to mitigate cloud network attacks an approach scalable to the sophisticated demands of 6G infrastructures.

Moreover, Yoon et al. [26] introduced a three-layer model to reduce MTD's operational costs in SDNs by identifying a subset of hosts to shuffle. They proposed a Greedy Backward Attack Path (BAP) prediction algorithm to trace vulnerable paths from attackers to critical servers and obfuscate hosts along these paths. The model employs attack graphs to assess vulnerability and optimize the shuffling process, making it particularly suitable for deployment in large-scale SDN and 5G-based networks.

Despite considerable progress, few approaches have effectively tackled MTD's cost-efficiency and the mitigation of DDoS attacks, particularly within SDN environments. Existing solutions often suffer from notable limitations some are tailored exclusively for cloud networks utilizing virtual machines. In contrast, others rely on complex game-theoretic or hash-based mechanisms that introduce significant computational overhead and latency at the SDN controller. To overcome these challenges, we propose enhancing the method introduced in Yoon et al. [26] (i.e., the BAP strategy) by incorporating the number of connections between hosts and critical servers. This modification is designed to improve the precision of DDoS attack mitigation while maintaining responsiveness and scalability in high-speed, dynamic 5G network settings.

Also, the Edge-based approach is one of the simplest and earliest methods used in modelling attack graphs. It assumes static connectivity and calculates vulnerability using edge weights. This technique has been discussed in basic

threat modeling frameworks, including MITRE ATT&CK and classical attack graph models [32]. The Backward Attack Propagation (BAP) method builds upon the principles of graph traversal and recursive impact analysis. It was inspired by backwards reachability concepts in network security and appears in works related to malware spread modelling and vulnerability prioritization [33]. The Edge Shuffling MTD strategy was introduced to dynamically disrupt attacker reconnaissance by periodically modifying the network's edge structure. Research has demonstrated that shuffling communication paths significantly delays attack success without major QoS penalties [34, 35]. IP Shuffling, or IP-level obfuscation, originated from early intrusion prevention systems and involves periodic reallocation of virtual address space to hosts, such as through IP hopping [27]. Server Relocation is another strategic MTD mechanism used in cloud and virtualized environments, wherein migrating services across logical nodes helps disrupt attack persistence [28]. Topology Randomization protects path prediction by generating stochastic topologies, effectively deterring lateral attacker movement [29]. Traffic Route Mutation leverages SDN programmability to reroute flows unpredictably, reducing exposure to repeated attacks, particularly in multi-tenant infrastructures [30]. Finally, Port Hopping, inspired by frequency-hopping in wireless communication, changes service ports periodically to evade predictable scanning tools, serving as a lightweight countermeasure against fixed-target attacks [31].

a) Positioning within Moving-Target Defenses.: Client–server shuffling and replica reassignment primarily leverage endpoint or pool volatility to dilute adversarial fixation; fast IP/flux switching perturbs addressing to increase reconnaissance and exploitation costs; live migration/partitioning remap compute or split flows to isolate blast radius. In contrast, our NoEC policy ranks hosts by a *cost-normalized edge-impact* score, $\Psi(h) = \deg_S(h)/c(h)$, and executes localized, budget-feasible reconfigurations that maximally *detach* host–server incidences per unit cost. This objective targets structural reachability of attack paths rather than address churn or workload relocation, yielding stronger path coverage when degree/cost heterogeneity is present, while reducing the operational footprint by avoiding global remapping.

b) Relation to Centrality and Cut-Based Formulations.: Our selection score $\Psi(h) = \deg_S(h)/c(h)$ can be viewed as a *cost-normalized degree centrality*. When costs are uniform ($c(h) \equiv 1$), ranking by Ψ coincides with degree centrality on the host side and thus favors hosts incident to many server edges. Relative to betweenness-centric views—which prioritize nodes traversed by many shortest paths—high-Ψ hosts in sparse or core–periphery bipartite topologies often lie on a large fraction of simple host–server walks, so detaching their incidences yields broad path coverage (cf. Fig. 8). From a cut/min-path perspective, our policy acts as a budgeted, local heuristic that maximizes *edges detached per unit cost* without solving a global cut on evolving demand. This is attractive when (i) the topology or costs change across shuffle epochs, (ii) the controller must keep per-epoch work near-linear, and (iii) path-length sensitivity is secondary to structural reachability. When path length or link capacity dominates (e.g., weighted shortest paths, congestion-sensitive defenses), classical betweenness/flow-based metrics or capacity-aware cuts may better reflect the operational objective; our framework remains compatible by incorporating such signals into $c(h)$ or by augmenting Ψ with weights.

III. Proposed Method (NoEC)

In distributed attacks such as DDoS, adversaries mobilize many vulnerable hosts and issue commands to synchronize these hosts to attack a specific target within a specified time interval. To minimize operational costs, adversaries strategically identify the fewest hosts needed to launch an attack successfully. As SDNs technologies offer higher speeds and better connectivity, adversaries will increasingly adapt their strategies to exploit these advancements.

The method proposed in this paper, Number of Edge Connections (NoEC), is designed to minimize implementation costs while maintaining or even increasing security levels. The complexity of NoEC algorithms directly affects their cost. To achieve these goals, NoEC is based on the BAP method of [26]. The BAP method identifies and disrupts hosts along the most vulnerable attack paths. However, it does not consider the distributed nature of modern attacks. In distributed DDoS attacks, the critical issue is the vulnerability of individual hosts and their potential role in the adversary's attack network. Therefore, NoEC modifies this approach by considering hosts that play a critical role in the enemy army, ensuring that the mixing process targets those that have the most significant impact on attack effectiveness.

a) Notation.: We uniformly use *mixing degree* d_i for host i and discard legacy synonyms (e.g., "clutter degree"). All figures, captions, and cross-references adopt this terminology consistently.

b) Reading the Running Example.: Figures 1 and 2 depict the bipartite host–server topology with hosts h_1–h_6 and servers s_1–s_5. In Fig. 1, intended adversarial traversals are highlighted in red. In Fig. 2, the same topology is shown after applying our policy: detached host–server incidences are rendered as dashed gray, illustrating how budget-feasible reconfiguration disrupts the highlighted paths.

We introduce an additional parameter, allowing us to achieve less complexity while maintaining or improving the results in many scenarios. Given the adversary's goal of amassing the smallest attack force possible and the nature of distributed attacks, we have designed a low-complexity MTD approach focusing on hosts with more connections to critical servers. In distributed attacks, the collective effect of a group of hosts is greater than that of individual hosts. Therefore, our approach prioritizes communication between critical hosts and servers rather than per-host costs.

Figure 3 shows an example that compares the proposed method (NoEC) and BAP.

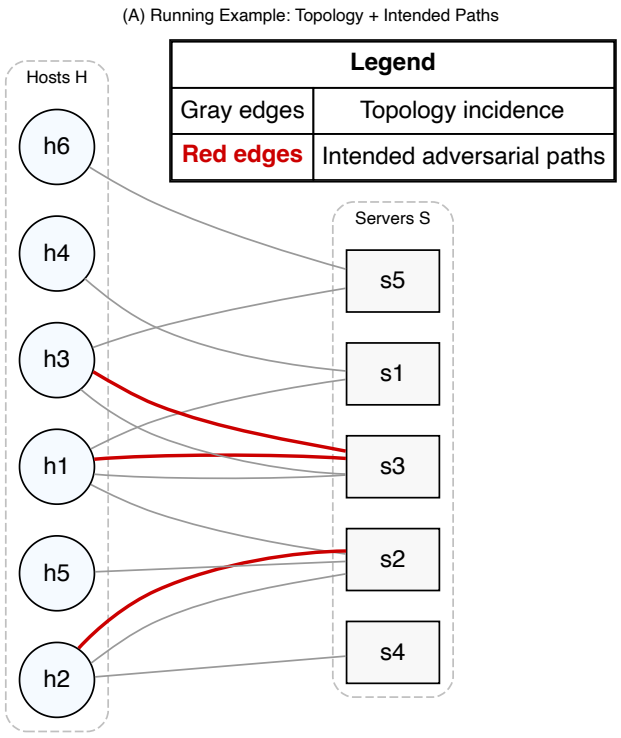


Fig. 1. (A) Running-example topology and intended adversarial paths (red).

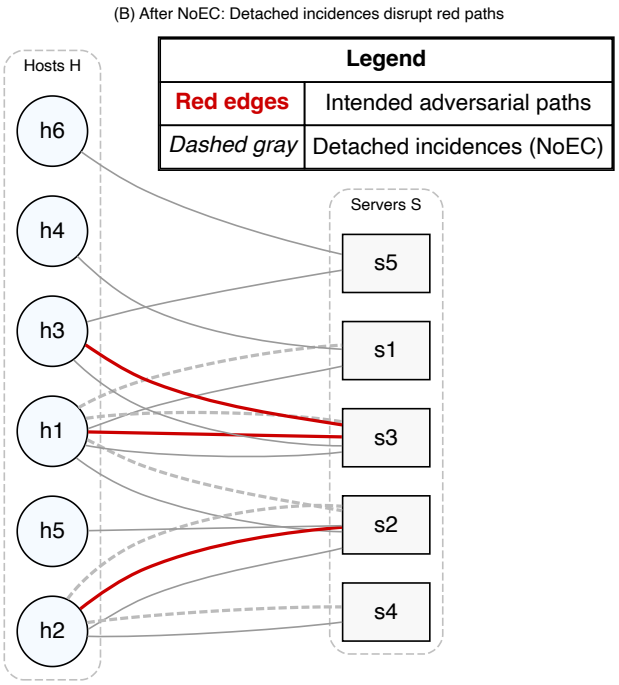


Fig. 2. (B) After NoEC: detached incidences (dashed gray) disrupting the red paths.

To illustrate the adversarial threat a three-stage process is depicted in Figure 4. In the first stage (Fig.4(a)), an external adversary attempts to reach target servers by leveraging multiple intermediate nodes with varying trust scores. These scores, typically learned or assigned by the SDN controller, represent the reliability or historical behavior of each node.

To mitigate the risk of exploitation, the SDN controller initiates a shuffling mechanism (Fig.4(b)), where nodes with lower trust values (e.g., N_5) are removed from the routing graph. This dynamic reconfiguration aims to reduce the number of vulnerable paths without disrupting legitimate traffic.

Despite these precautions, the third stage (Fig.4(c)) shows a scenario in which the adversary successfully bypasses defenses via a remaining weak node (e.g., N_4 with a trust score of 0.1), eventually compromising the target server s_2. This highlights the importance of adaptive and continuous trust management to ensure adequate defense in SDN environments.

In our threat model, node values represent the cost of compromising each host and launching a DDoS attack on critical servers. While the BAP method focuses on minimizing the compromise cost of individual hosts, NoEC shifts the priority to connections between hosts and critical servers. This fundamental distinction highlights a key advantage of NoEC: by targeting highly connected nodes, it significantly reduces the feasibility of successfully executing DDoS attacks across all critical servers. In contrast, BAP's emphasis on localized cost assessments can overlook broader network dynamics, potentially allowing attacks to succeed. This comparison is illustrated in Figure 5 and Figure 6, which underscore the superior defensive coverage provided by NoEC.

We consider that the attacker's goal is to launch a DDoS attack that targets multiple critical servers simultaneously. This scenario can be modelled as $\mathcal{N} = (S, \mathcal{C})$, where S represents the total number of critical servers and $\mathcal{C}$ is an ordered set of host compromises. Specifically, the costs are $\mathcal{C} = \{c_1, c_2, \ldots, c_H\}$, with c_i representing the minimum cost required to compromise the i^{th} host (h_i) by the enemy and H indicates the total number of hosts.

In this model, each critical server must be protected by considering individual costs and the network connectivity structure. The complexity and interconnectedness of these advanced networks mean that an adversary can exploit multiple potential entry points and use extensive network connectivity to power their attack.

We define a clutter degree for each host, denoted by d_i. Host mixing degree i^{th}, d_i, is calculated as the ratio of the number of servers directly connected to that host to the total number of servers. This metric provides a measure of host connectivity within the network. We hypothesize that shuffling hosts based on this degree performs better in mitigating attacks than shuffling based on the compromised cost of each host. This approach leverages SDNs' ability to manage configurations for real-time adjustments based on congestion levels dynamically.

An example network, $\mathcal{N}_\mathcal{E}$, is shown in Figure 5 and Figure 6. ($\mathcal{N}_\mathcal{E}$ topology with two critical servers and several common hosts).

For this network, the set of compromised costs is $\mathcal{C}_\mathcal{E} = \{\infty, 0.7, 0.6, 0.2, 0.1, 0.5\}$. From this, we can infer that the first host, h_1, is invulnerable to DDoS attacks targeting crit-

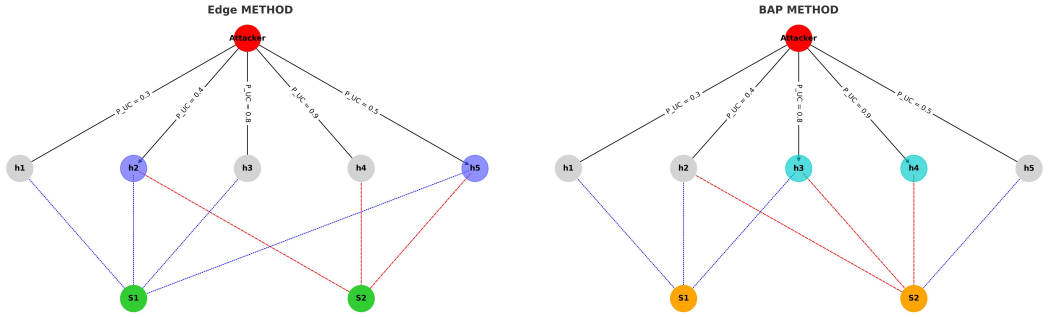


Fig. 3. The figure illustrates Host Nodes by cost in the BAP solution and Critical Edge Nodes arranged by connections in the Edge method. Attack Nodes represent sources of DDoS attacks. Cost-Based Links connect hosts according to the BAP solution, while Connection-Based Links associate critical edge nodes in the Edge method.

ical servers. Moreover, the compromise costs for hosts h_3, h_5 and h_6 are equal. The degrees of mixing of the hosts is $\{1/7, 2/7, 1/7, 1/7, 2/7\}$. or in another example, for this network configuration, the set of compromise costs is defined as

$$\mathcal{C}_{\mathcal{F}} = \{\infty, 1.5, 1.0, 1.3, 1.0, 1.1\}.$$

Analyzing this data reveals that the first host, h_1, is impervious to DDoS attacks aimed at high-value servers. Additionally, the compromise costs for hosts h_3 and h_4 are identical, suggesting similar levels of vulnerability in these nodes. The shuffling degrees for each host, based on their connection profiles, are as follows:

$$\left\{\frac{1}{10}, \frac{1}{10}, \frac{3}{10}, \frac{1}{10}, \frac{3}{10}, \frac{1}{10}\right\}.$$

The BAP method suggests selecting the most vulnerable hosts for shuffling, which include h_3, h_5 and h_6. However, s_1 still has three unblocking connections, which requires hitting another host connected to s_1. Considering that h_2 has the lowest compromise cost, it is selected in the next step. Therefore, the set of hosts to hit $\{h_2, h_3, h_5, h_6\}$ becomes 3.9 at a total cost. Another example is the BAP method, which suggests selecting the most vulnerable hosts for shuffling, which include h_2, h_4, and h_7. However, s_2 still has two remaining unblocked connections, necessitating an additional selection of a host connected to s_2. Since h_1 has the lowest compromise cost, it is chosen in the next step. Consequently, the set of hosts to target, $\{h_1, h_2, h_4, h_7\}$, results in a total compromise cost of 4.2.

In contrast, NoEC selects hosts based on their degrees of confusion. Therefore, the hosts with the highest mixing degrees are merged: $\{h_3, h_4, h_5\}$. This set of hosts effectively mitigates the attack because s_1 and s_2 have fewer than three unblocking connections. The total cost of this collection is 3.21. As a result, NoEC identifies a set of low-cost hosts for relocation compared to BAP. Furthermore, in another example, the Minimum Shuffling Costs (MSC) method of BAP selects hosts based on their highest shuffling degrees. Thus, the hosts with the most significant mixing levels are grouped: $\{h_1, h_6, h_8\}$. This group of hosts successfully counters the attack, as s_2 and s_4 retain only two active connections each, falling below the threshold. The total compromise cost for this group is 2.85. Consequently, the MSC method identifies a cost-effective set of hosts for reallocation compared to the BAP approach.

IV. Mathematical Formulation and Proof of Effectiveness of NoEC

The Figure 7 presents a realistic DDoS attack scenario in a network composed of attacker nodes (A), hosts (h), and a critical server (s). This structure shows how adversaries can reach high-value servers by compromising a small subset of highly connected hosts. The NoEC algorithm aims to identify and relocate those hosts with the highest number of connections to critical servers (i.e., with the highest d_i), thereby effectively disrupting potential attack paths. In this model, the objective function minimizes the aggregated compromise costs c_i, subject to a constraint that ensures sufficient disconnection of critical servers from potential attack vectors.

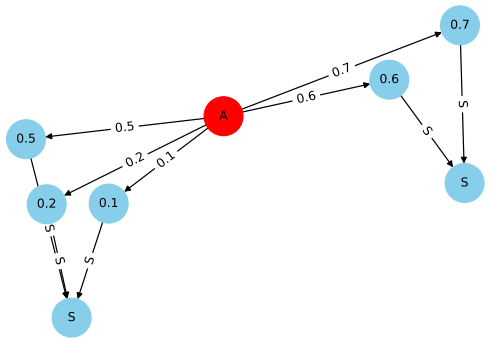
A. Problem Setting

Let $H = \{h_1, h_2, \ldots, h_n\}$ be the set of hosts and $S = \{s_1, s_2, \ldots, s_m\}$ the set of critical servers. Define the bipartite attack graph $\mathcal{G} = (H \cup S, E)$ where edges $(h_i, s_j) \in E$ indicate connectivity. Each host h_i has a compromise cost $c_i \in \mathbb{R}_{\geq 0} \cup \{\infty\}$. The adversary aims to compromise a minimal-cost subset of hosts to reach all critical servers.

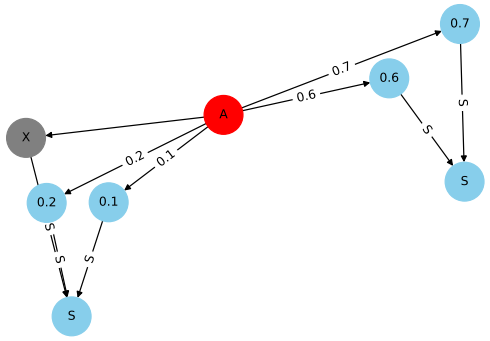
The goal of MTD is to identify and shuffle a subset of hosts $\mathcal{H} \subseteq H$ such that the adversary cannot maintain enough access to attack any server.

B. Key Definitions

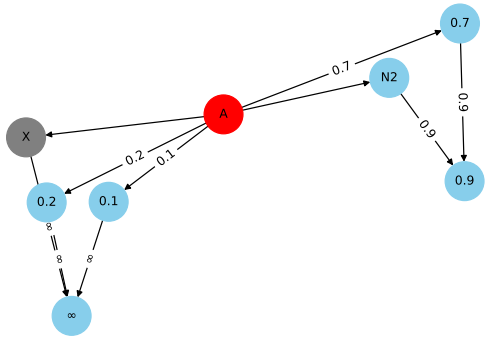
- **Host Degree:** $d_i = \frac{1}{|S|} \sum_{j=1}^{m} \mathbb{1}_{(h_i, s_j) \in E}$.
- **Server Degree Post-Shuffling:** Let $\delta_j(\mathcal{H}) = |\{h_i \notin \mathcal{H} : (h_i, s_j) \in E\}|$.
- **Attack Threshold:** $\tau \in \mathbb{Z}_{\geq 0}$ is the maximum allowed number of unshuffled connections per server for security to hold.



(a) Initial State



(b) After Shuffling



(c) After Attacking

Fig. 4. Three-stage process of trust-based defense in SDN. (a) The adversary can access multiple paths through intermediate nodes with different trust scores. (b) The controller removes low-trust paths to mitigate risk. (c) Despite shuffling, the attacker successfully exploits a remaining vulnerable node.

C. Optimization Objective

We aim to solve the constrained optimization problem:

$$\min_{\mathcal{H}\subseteq H} \sum_{h_i\in\mathcal{H}} c_i, \tag{P1}$$

$$\text{s.t.} \quad \delta_j(\mathcal{H}) \le \tau, \quad \forall j = 1,\dots,m. \tag{C1}$$

The above model (P1) is the foundation of BAP, which focuses on host cost minimization.

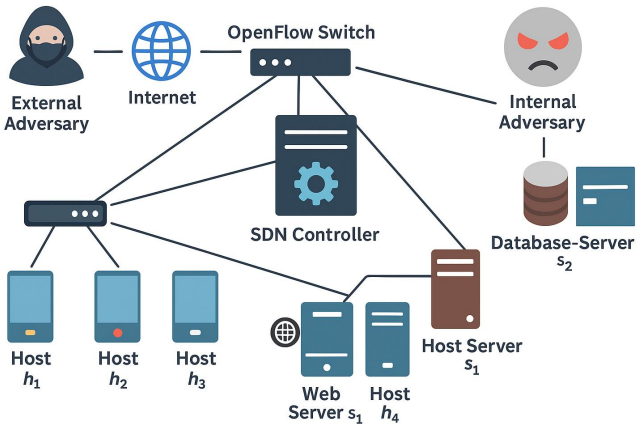


Fig. 5. Illustration of the SDN-based network architecture with internal and external adversaries. The SDN controller centrally manages OpenFlow switches and orchestrates data forwarding across connected hosts and servers. The external adversary attempts to penetrate the network from the internet, while the internal adversary targets critical assets such as the database server s_2. The controller continuously monitors and updates the forwarding rules to maintain security and efficiency.

D. NoEC Reformulation

We redefine the selection problem using the normalized degree d_i to capture host importance:

$$\max_{\mathcal{H}\subseteq H} \sum_{h_i\in\mathcal{H}} d_i, \tag{P2}$$

$$\text{s.t.} \quad \delta_j(\mathcal{H}) \le \tau, \quad \forall j = 1,\dots,m, \tag{1}$$

$$\sum_{h_i\in\mathcal{H}} c_i \le B. \tag{C2}$$

Here, B is the MTD defense cost. The NoEC strategy maximises connectivity disruption at the same or lower cost.

E. Theoretical Comparison and Proof

Theorem. Under equal cost constraints, the NoEC strategy removes more host-server connections than BAP. Therefore, for any feasible cost B and threshold τ, we have:

$$|\{s_j \in S : \delta_j(\mathcal{H}_{\text{NoEC}}) \le \tau\}| \ge |\{s_j \in S : \delta_j(\mathcal{H}_{\text{BAP}}) \le \tau\}|.$$

Proof. Let $\mathcal{H}_{\text{BAP}}$ and $\mathcal{H}_{\text{NoEC}}$ denote the sets of hosts selected by BAP and NoEC, respectively, under the same cost constraint:

$$\sum_{h_i\in\mathcal{H}_{\text{BAP}}} c_i = \sum_{h_i\in\mathcal{H}_{\text{NoEC}}} c_i \le B.$$

Define the connectivity disruption metric:

$$\Psi(\mathcal{H}) := \sum_{s_j\in S} [\deg(s_j) - \delta_j(\mathcal{H})] = \sum_{h_i\in\mathcal{H}} |\{s_j : (h_i, s_j) \in E\}|.$$

This is equivalent to:

$$\Psi(\mathcal{H}) = \sum_{h_i\in\mathcal{H}} d_i \cdot |S|.$$

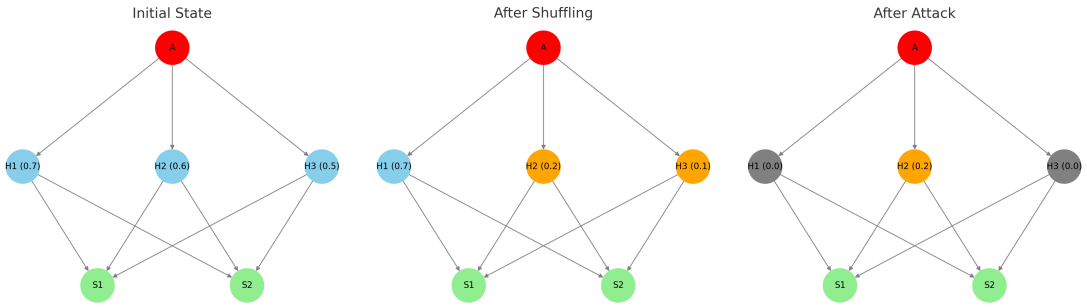
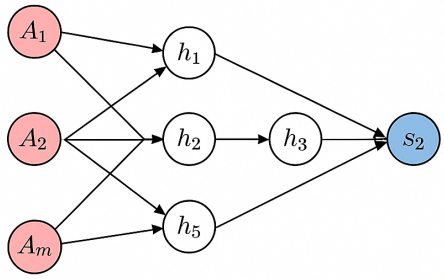


Fig. 6. Three-stage illustration of the trust-aware defense mechanism in SDN. Initially, the adversary (A) can access all intermediary nodes with different trust scores. In the second stage, the SDN controller applies a shuffling mechanism to eliminate low-trust nodes from routing. In the third stage, the attacker adjusts its path and successfully exploits a vulnerable connection, demonstrating that dynamic reconfiguration must be coupled with trust evaluation to prevent successful breaches.



$$\min \sum_{i \in H}^{n} \sum c_i x_i,$$

$$\text{cons.} \sum_{i\ H} d_i x_i \geq S,$$

$$x_i \in |0,1| \quad \forall i \in H.$$

Fig. 7. Network-based formulation of DDoS mitigation using host mixing degree d_i and cost c_i. Attackers (A) attempt to reach the critical server (s_2) via multiple hosts. The optimization minimizes cost while breaking key connections.

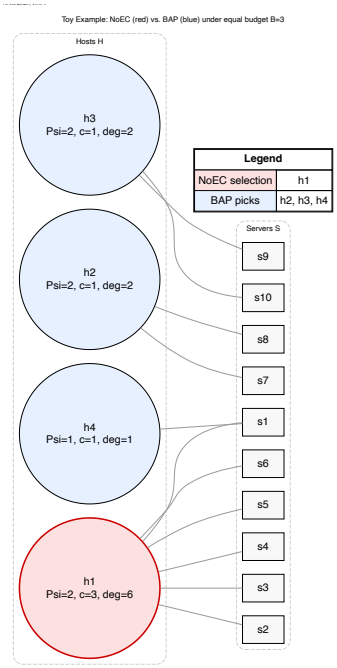


Fig. 8. Toy example where NoEC's high-Ψ host detaches more edges than multiple low-cost hosts (BAP) under the same budget $B = 3$.

By construction of $\mathcal{H}_{\text{NoEC}}$, we have:

$$\Psi(\mathcal{H}_{\text{NoEC}}) \geq \Psi(\mathcal{H}_{\text{BAP}}).$$

Thus, NoEC removes more connections per unit cost. Since server compromise depends on maintaining more than τ connections, and NoEC causes a greater drop in δ_j across $s_j \in S$, it ensures more servers satisfy the condition $\delta_j \leq \tau$.

Hence,

$$A_{\text{NoEC}} := |\{s_j \in S : \delta_j(\mathcal{H}_{\text{NoEC}}) \leq \tau\}|$$
$$\geq A_{\text{BAP}} := |\{s_j \in S : \delta_j(\mathcal{H}_{\text{BAP}}) \leq \tau\}|,$$

which proves that NoEC is strictly more effective or equally effective under the same cost. □

F. Complexity Analysis

Let $|E|$ be the number of edges in $\mathcal{G}$. The complexity of computing d_i for all $h_i \in H$ is $\mathcal{O}(|E|)$. Sorting hosts by d_i and selecting within cost B yields a total complexity of $\mathcal{O}(n \log n + |E|)$, which is comparable to BAP.

G. Intuition and Toy Example: NoEC vs. BAP

We complement the formal result with a compact, visual intuition. As illustrated in Fig. 8, under the same budget a single high-impact host chosen by NoEC—ranked by the cost-normalized score $\Psi(h) = \deg_S(h)/c(h)$—can detach more host–server edges than several low-cost hosts chosen by BAP. This gap grows in heterogeneous regimes where degree and cost are skewed.

a) Toy instance.: Consider a bipartite graph $G = (H \cup S, E)$ with per-host reconfiguration cost $c(h) > 0$ and server-degree $\deg_S(h)$. Let $H = \{h_1, h_2, h_3, h_4\}$ and $S = \{s_1, \ldots, s_{10}\}$ with $\deg_S(h_1) = 6$, $c(h_1) = 3$; $\deg_S(h_2) = 2$,

Algorithm 1 NoEC: Cost-Normalized Edge-Detachment Selection

Require: Bipartite graph $G = (H \cup S, E)$; per-host costs $c(h) > 0$; budget $B > 0$

Ensure: Selected host set $\mathcal{H}_k$; detached incidence set $\mathcal{E}_{\text{det}}$

$\mathcal{H}_k \leftarrow \emptyset,\ \mathcal{E}_{\text{det}} \leftarrow \emptyset,\ B_{\text{rem}} \leftarrow B$ ▷ Initialize

for all $h \in H$ **do** $\deg_S(h) \leftarrow |\{(h,s) \in E\}|;\ \Psi(h) \leftarrow \deg_S(h)/c(h)$ **end for**

Build max-heap $\mathcal{Q}$ keyed by $\Psi(h)$ over H ▷ $\mathcal{O}(|H|)$ build

while $\mathcal{Q} \neq \emptyset$ **and** $B_{\text{rem}} > 0$ **do**

  $h^\star \leftarrow \textsc{PopMax}(\mathcal{Q})$ ▷ $\mathcal{O}(\log |H|)$

  if $c(h^\star) \le B_{\text{rem}}$ **then**

    $\mathcal{H}_k \leftarrow \mathcal{H}_k \cup \{h^\star\};\ B_{\text{rem}} \leftarrow B_{\text{rem}} - c(h^\star)$

    $\mathcal{E}_{\text{det}} \leftarrow \mathcal{E}_{\text{det}} \cup \{(h^\star, s) \in E\}$ ▷ Detach all incidences of $h^\star$

  else if $B_{\text{rem}}/c(h^\star) \in (0,1)$ **and** partial detachment allowed **then**

    Detach up to B_{rem} worth of $(h^\star, s)$, ordered by edge impact; set $B_{\text{rem}} \leftarrow 0$

return $\mathcal{H}_k,\ \mathcal{E}_{\text{det}}$

Tie-breaking. If Ψ ties, prefer larger $\deg_S(h)$; then smaller $c(h)$; then a stable ID.

Per-epoch complexity. Discovery/refresh $\mathcal{O}(|E|)$; selection $\mathcal{O}(|H| \log |H|)$; rule updates $\mathcal{O}\big(k + \sum_{h \in \mathcal{H}_k} \deg_S(h)\big)$.

$c(h_2) = 1$; $\deg_S(h_3) = 2$, $c(h_3) = 1$; $\deg_S(h_4) = 1$, $c(h_4) = 1$; and $B = 3$. Then

$$\Psi(h_1) = \tfrac{6}{3} = 2, \quad \Psi(h_2) = \Psi(h_3) = 2, \quad \Psi(h_4) = 1.$$

BAP (cheapest-first) removes $\{h_2, h_3, h_4\}$ and detaches $2 + 2 + 1 = 5$ edges, whereas NoEC selects h_1 and detaches 6 edges with the same B (see Fig. 8). The advantage widens as incidence and costs become more skewed.

Algorithmic Overview and Pseudocode

We now summarize the selection procedure that operationalizes our cost-normalized objective. The inputs are a bipartite topology $G = (H \cup S, E)$, per-host reconfiguration costs $c(h) > 0$, and a per-epoch budget $B > 0$. Each host $h \in H$ is scored by

$$\Psi(h) = \frac{\deg_S(h)}{c(h)},$$

where $\deg_S(h)$ is the number of incident host–server edges. The algorithm returns a selected host set $\mathcal{H}_k$ (budget-feasible) and the detached incidence set $\mathcal{E}_{\text{det}} = \{(h,s) \in E : h \in \mathcal{H}_k\}$. We adopt a max-heap keyed by $\Psi(\cdot)$ for near-linear per-epoch cost (cf. Sec. VIII-D); ties prefer larger $\deg_S(h)$, then smaller $c(h)$, then a stable ID. An optional "partial detachment" mode allows using the remaining fractional budget on a subset of incidences of the last-picked host, ordered by edge impact, when strict per-host indivisibility is not required.

b) Assumptions and Proof Sketch.: We specify sufficient conditions under which the cost-normalized selection $\Psi(h) = \deg_S(h)/c(h)$ weakly dominates a cheapest-first baseline (BAP) in terms of total detached incidences.

Setup. Let $G = (H \cup S, E)$ be bipartite with per-host cost $c(h) > 0$ and server-incidence $\deg_S(h)$. Given budget $B > 0$, define for any selection $X \subseteq H$ the value $V(X) = \sum_{h \in X} \deg_S(h)$ and the cost $C(X) = \sum_{h \in X} c(h) \le B$.

Special case (uniform costs). If $c(h) \equiv c_0$, then maximizing $V(X)$ under $C(X) \le B$ reduces to picking the $k = \lfloor B/c_0 \rfloor$ hosts with largest $\deg_S(\cdot)$. NoEC ranks by $\Psi(h) = \deg_S(h)/c_0$ and thus coincides with degree-order selection, which dominates any cheapest-first policy (since all are equally cheap). Hence $V(X_{\text{NoEC}}) \ge V(X_{\text{BAP}})$.

Monotone score–cost alignment (sufficient condition). Suppose for any $i, j \in H$,

$$c(i) \le c(j) \implies \Psi(i) \le \Psi(j),$$

i.e., cheaper hosts do not have strictly higher cost-normalized impact than more expensive ones. Then any cheapest-first selection can be transformed into a NoEC-prefix selection without exceeding budget while not decreasing $V(\cdot)$.

Proof sketch (exchange argument). Order hosts by nonincreasing Ψ: $h_1, \ldots, h_n$ with $\Psi(h_1) \ge \cdots \ge \Psi(h_n)$. Consider any BAP-feasible set X, and let $h_t \notin X$ be the first (highest-Ψ) host omitted by X. Since X is cheapest-first and costs are aligned with Ψ by assumption, there exists $h' \in X$ with $c(h') \ge c(h_t)$ and $\Psi(h') \le \Psi(h_t)$. Exchanging h' with h_t keeps cost nonincreasing (C does not increase) and increases or preserves value:

$$V\big((X \setminus \{h'\}) \cup \{h_t\}\big) - V(X) = \deg_S(h_t) - \deg_S(h') \ge 0,$$

because $\Psi(h_t) \ge \Psi(h')$ and $c(h_t) \le c(h')$. Repeating the exchange yields a prefix of the Ψ-ordering (the NoEC set) with V no smaller than BAP's. Thus $V(X_{\text{NoEC}}) \ge V(X_{\text{BAP}})$ under the stated condition. □

Remarks. (i) The alignment condition is *sufficient* but not necessary; in practice, NoEC often outperforms BAP even when costs and scores are only partially aligned (cf. Fig. 8). (ii) When costs are uniform, the result is immediate (degree-order selection). (iii) If partial detachment is allowed, the same exchange logic applies to edge-level units by selecting highest-impact incidences first.

H. Approximation Properties

We formalize the structure of our objective and the implications for greedy selection.

a) Objective as Coverage.: Let $\mathcal{E}_{\text{det}}(X) = \{(h,s) \in E : h \in X\}$ be the set of detached incidences induced by a host set $X \subseteq H$. Define the (incidence) coverage value

$$F(X) := |\mathcal{E}_{\text{det}}(X)| = \sum_{h \in X} \deg_S(h) - \big|\text{overlaps among } \{\mathcal{E}_{\text{det}}(\{h\})\}_{h \in X}\big|.$$

Equivalently, when evaluating *path* coverage, let $\mathcal{U}(X) = \bigcup_{h \in X} \mathcal{P}(h)$ be the union of simple host–server paths intersecting detached incidences; then $F(X) = |\mathcal{U}(X)|$ up to a normalization by $|\mathcal{P}|$ (Sec. VIII-H).

Proposition IV.1 (Monotonicity and submodularity)**.** *The set function $F : 2^H \to \mathbb{R}_{\geq 0}$ defined above is* monotone *and* submodular. *In particular, for any $A \subseteq B \subseteq H$ and $h \in H \setminus B$,*

$$F(A \cup \{h\}) - F(A) \;\geq\; F(B \cup \{h\}) - F(B),$$

and $F(A) \leq F(B)$.

Proof sketch. F counts the cardinality of a union of per-host incidence/path sets. Unions of sets yield a classical coverage function, which is monotone (adding sets does not reduce the union) and submodular (marginal gains decrease as the union grows), i.e., diminishing returns. □

b) Cardinality constraint.: When costs are uniform ($c(h) \equiv c_0$) and we select exactly k hosts, the standard greedy that adds the host with maximum marginal gain $\Delta(h \mid X)$ at each step achieves a $(1 - 1/e)$-approximation of the optimal coverage. This follows directly from Proposition IV.1 and classical results for monotone submodular maximization under a cardinality constraint.

c) Budget (knapsack) constraint.: With heterogeneous costs $c(h)$ and total budget B, we employ the cost-normalized greedy (ratio-greedy) that repeatedly selects the host maximizing $\Delta(h \mid X)/c(h)$ while budget remains. A light-weight guarantee is obtained by comparing the ratio-greedy solution with the best single host ("best-singleton" patch) and returning the better of the two; this yields a constant-factor bound for budgeted coverage while preserving controller efficiency. Stronger $(1 - 1/e)$-type guarantees are attainable via slightly heavier variants (e.g., partial enumeration or continuous-greedy), which we eschew here to keep per-epoch overhead near-linear (Sec. VIII-D).

d) Practical note.: Our implementation follows the ratio-greedy with a max-heap keyed by $\Psi(h) = \deg_S(h)/c(h)$ and incorporates the best-singleton check. Empirically, its coverage closely tracks stronger baselines while keeping selection time $\mathcal{O}(|H| \log |H|)$ and rule activity bounded (Sec. VIII-E).

I. Capacity- and Path-Aware Variants

Our score can incorporate resource and path signals through either (i) a *weighted incidence* on the numerator or (ii) an *effective cost* on the denominator.

J. Trust Signal and Temporal Decay

We maintain a bounded trust score $T(h) \in [0, 1]$ per host h, where larger values indicate higher confidence in the host's integrity. Trust is updated online from alerts/telemetry and used only through cost shaping to avoid destabilizing the selection.

a) Evidence and normalization.: Let $e(h, t) \in [0, 1]$ denote the normalized evidence at epoch t from NIDS logs, rate-limiters, or anomaly scores (0: no concern; 1: strong concern). We map raw detectors to $e(\cdot)$ by min–max or percentile normalization per detector and then average across sources (weights can be uniform unless otherwise specified).

b) Temporal update (EMA with floors/ceilings).: With decay parameter $\lambda \in (0, 1]$ and bounds $0 < T_{\min} \leq T_{\max} \leq 1$, we update

$$T_{t+1}(h) \;=\; \mathrm{clip}_{[T_{\min}, T_{\max}]}\Big(\big(1 - \lambda\big)\, T_t(h) \;+\; \lambda\,\big(1 - e(h, t)\big)\Big),$$

initialized at $T_0(h) = T_{\max}$. The EMA smooths bursty signals; $T_{\min}, T_{\max}$ prevent saturation (defaults below).

c) Cooldown to avoid oscillations.: If $e(h, t) \geq \theta_{\text{high}}$ for C consecutive epochs, we assert a cooldown: mark h as *cooling* for L_{cool} epochs during which $T(h)$ cannot increase (only decay applies). This avoids rapid flip-flops near decision boundaries.

d) Integration via effective cost.: We shape the denominator in the score using

$$c'(h) = c(h)\,\big(1 + \gamma\, r(h)\big),$$
$$r(h) = \beta\big(1 - T(h)\big) + \eta\, \mathrm{lat\_sens}(h).$$

with $\gamma, \beta, \eta \geq 0$ and $\mathrm{lat\_sens}(h) \in \{0, 1\}$ a policy flag for ultra-low-latency slices. Selection uses $\Psi(h) = \deg_S(h)/c'(h)$ (or $\widetilde{\deg}_S(h)/c'(h)$ for weighted variants). Because $c'(h) > 0$ and weights are nonnegative, the induced coverage remains a (weighted) union; the monotone-submodular structure and the approximation properties in Sec. IV-H continue to hold.

e) Stability and parameter defaults.: We choose $\lambda = 0.2$, $(T_{\min}, T_{\max}) = (0.1, 1.0)$, $\theta_{\text{high}} = 0.8$, $C = 2$, $L_{\text{cool}} = 3$, $\beta = 1.0$, and $\eta = 1.0$ unless stated otherwise. For the jitter in Sec. VIII-E, we keep $\gamma \in \{0, 0.25, 0.5\}$ and report sensitivity in the supplement. These values bias the policy away from persistently low-trust hosts without materially changing coverage when alerts are rare or weak.

f) Reproducibility.: We log $(\lambda, T_{\min}, T_{\max}, \theta_{\text{high}}, C, L_{\text{cool}}, \beta, \eta, \gamma)$ and the PRNG seeds for any stochastic detectors. The same trust pipeline (normalization and EMA) is reused across methods to ensure fairness.

g) Weighted incidence.: Let $w_{hs} \geq 0$ encode per-incidence importance (e.g., capacity-, demand-, or latency-aware). Replace the server-incidence by

$$\widetilde{\deg}_S(h) \;:=\; \sum_{(h,s) \in E} w_{hs}, \qquad \Psi_w(h) \;:=\; \frac{\widetilde{\deg}_S(h)}{c(h)}.$$

When $w_{hs} \in [0, 1]$ are nonnegative, maximizing the weighted coverage remains a union-of-sets objective (now with weights), which is monotone submodular; thus, the approximation properties in Sec. IV-H continue to hold (cardinality: $(1 - 1/e)$; budget: ratio-greedy with the best-singleton patch).

h) Effective cost shaping.: QoS risk or operational friction can be folded into the denominator by defining

$$c'(h) \;:=\; c(h)\cdot\big(1+\gamma\, r(h)\big),$$

where $r(h)\in[0,1]$ aggregates risk (e.g., latency sensitivity, slice policy, trust factor) and $\gamma\ge 0$ tunes its influence. The selection then uses

$$\Psi_w(h) := \frac{\widetilde{\deg}_S(h)}{c'(h)},$$

$$\Psi(h) := \frac{\deg_S(h)}{c'(h)} \quad \text{(if only cost shaping is desired).}$$

This preserves the selection mechanics while biasing the policy away from high-risk hosts.

i) Practical instantiation.: A typical choice is

$$w_{hs} = \underbrace{\frac{\mathrm{cap}(h,s)}{\max_{(u,v)\in E}\mathrm{cap}(u,v)}}_{\text{capacity share}} \cdot \underbrace{\frac{\mathrm{demand}(h,s)}{\sum_{(u,v)\in E}\mathrm{demand}(u,v)}}_{\text{demand norm}} \cdot \underbrace{\frac{1}{1+\lambda\,\mathrm{hoplen}(h\to s)}}_{\text{path-length discount}}.$$

with $\lambda\ge 0$. For cost shaping, let $r(h)=\beta\big(1-T(h)\big)+\eta\,\mathrm{lat\_sens}(h)$ where $T(h)\in[0,1]$ is a trust score and $\mathrm{lat\_sens}(h)\in\{0,1\}$ flags ultra-low-latency slices.

j) Reporting and ablation.: We report both unweighted and weighted variants in Sec. VIII-H and sweep λ,γ (and β,η when enabled); coverage and QoS indices are shown with 95% CIs. Empirically, moderate capacity/path weighting improves disruption under congestion while preserving selection runtime.

V. System Architecture

We have developed a system within a SDN environment to implement the NoEC strategy. This system consists of four primary components: *critical servers*, *typical hosts*, *network devices*, and an *SDN controller*. Critical servers represent high-value assets that network administrators aim to protect from DDoS attacks. In contrast, typical hosts are vulnerable endpoints that adversaries may compromise to form a botnet for launching such attacks. These hosts and servers are interconnected via network devices, specifically, OpenFlow-enabled switches. The SDN controller plays a central role by issuing forwarding rules and control messages to manage these network devices, as illustrated in Figure 5.

To operationalize NoEC, the SDN controller incorporates five functional modules:

- **Network Topology Discovery (NTD)** – Responsible for gathering real-time network structure and connectivity data.
- **Shuffling Degree Calculator (SDC)** – Evaluates the number of connections between hosts and critical servers to determine shuffling priorities.
- **Impact Assessment System (IAS)** – Estimates the potential threat each host poses based on connectivity metrics.

Algorithm 2 SDC Module: Computing Shuffling Degrees

```
Initialize ne[i] ← 0 for all i ∈ {1, ..., H}      ▷ Connections count per host
Initialize total ← 0                               ▷ Tracks total host-to-server links
for i ← 1 to H do                                  ▷ Loop over hosts
    for j ← 1 to S do                              ▷ Loop over critical servers
        if conn(i, j) = 1 then                     ▷ If host h_i connected to server s_j
            ne[i] ← ne[i] + 1
            total ← total + 1
for i ← 1 to H do
    d[i] ← ne[i]/total                             ▷ Normalize degree
return d                                           ▷ Return shuffling degree vector
```

Fig. 9. The SDC module evaluates the importance of each host by calculating how frequently it connects to critical servers. This connection count is then normalized across the network to assign a probability-based shuffling degree to each host. Hosts with higher connectivity are assigned higher degrees, making them more likely to be selected for shuffling during an attack.

- **Shuffling Impact Determiner (SID)** – Determines which nodes to shuffle based on assessed risk and cost-efficiency.
- **Forwarding Engine (FEG)** – Enforces selected shuffling actions by updating forwarding rules on network devices.

A. Network Topology Discoverer (NTD)

The **Network Topology Discovery (NTD)** module utilizes the OpenFlow Discovery Protocol (OFDP) to ascertain the current state and topology of the network. This process involves identifying various network nodes and their interconnections, thereby enabling the generation of $\mathcal{C}$. The network administrator provides information regarding host vulnerabilities, interdependencies, and a predefined list of critical servers. Based on this input, the NTD module constructs the network model, denoted as $\mathcal{N}$, and forwards it to the **Shuffling Degree Calculator (SDC)** module. This procedure is triggered during network initialization, when the complete topology is discovered and communicated to the SDC module. The NTD module is designed to accurately capture the network state and structure, even in high-density and high-speed environments, thus facilitating effective NoEC implementation and ensuring robust defense against advanced distributed attacks.

B. Shuffling Degree Calculator (SDC)

The SDC module determines each host's congestion degree in the network. It receives the NTD module's network model and calculates each host's entanglement degrees. The mixing degree, d_i, is calculated for each host i using the connectivity information provided in $\mathcal{C}$. The algorithm implemented by the SDC module is provided in Algorithm 2 and Algorithm 1. After the calculation, the list of scrambling degrees is sent to the SID module.

The SDC module adjusts to conditions by providing accurate shuffling degree calculations to help prevent DDoS attacks and ensure network integrity and performance. It uses OFDP to assess the network's state, identifying vulnerabilities and key server connections. The calculated shuffling degrees help prioritize hosts at high risk of being targeted.

Algorithm 3 SID Module: Selecting Hosts for Shuffling

```
Initialize Top ← top μ + ρ hosts with highest d[i] values ▷ Pre-selected influential hosts
for each shuffling interval do
    Λ ← empty list ▷ Hosts selected for shuffling
    for i ← 1 to H do
        r ← Random(0, 1)
        if r < d[i] then
            Append i to Λ
    Send Λ to FEG for flow rule updates
```

Fig. 10. The SID module selects hosts for shuffling based on their previously calculated shuffling degrees. During each shuffling cycle, a random number is generated for each host and compared with its shuffling degree. If the random number is smaller, the host is marked for shuffling. This probabilistic method ensures dynamic and cost-effective defense by focusing more on high-risk hosts.

C. Shuffling Implementation and Decision (SID)

The SID module uses the NoEC algorithm to select hosts for shuffling, based on information from the SDC module. Reconfiguration and shuffling start at each fixed interval of σ seconds. Hosts are chosen for shuffling during each scrambling interval based on their degree. Specifically, each host h_i has a clutter probability d_i, where d_i represents its degree. This probabilistic approach ensures that hosts with higher hybrid degrees (i.e., those more critical to defending against attacks) are moved more frequently, aligning with their importance in mitigating potential DDoS threats.

A flow entry timeout triggers the SID module to detect the start of a new scrambling interval. This timeout is detected via the OpenFlow message type *OFPT_FLOW_REMOVED*, indicating the expiration of flow entries. Upon receiving this notification, the SID module evaluates the type of the current interval and determines the set of hosts, denoted as λ, that should be scrambled in this interval.

The set λ is then passed to the FEG (Flow Entry Generator) module, configuring the flow entries necessary to perform the blending. This systematic approach ensures that host migration is performed efficiently and NoEC-compliant, effectively protecting against evolving DDoS threats in dynamic 5G/6G network environments. The algorithm of the SID module is shown in Algorithm 3 and Algorithm 4.

D. IP Address Assignment and Shuffling (IAS)

The IAS module maintains a pool of IP addresses within the designated network address space, where each address is linked to a status flag indicating its current availability. During a shuffling process, when hosts require new IP addresses, the IAS module selects a random address from the network where the flag is unset, indicating the address is currently available. Once selected, this address is assigned to the host, and its flag is updated to reflect that it is now in use. The newly assigned IP addresses are then communicated to the FEG (Flow Entry Generator) module for implementation in the network.

Algorithm 4 Host Evaluation and Shuffling Strategy (NoEC)

Require: Connection matrix C of size $H \times S$, number of hosts H, number of servers S, shuffling interval σ, total simulation time T
Ensure: List of hosts to be shuffled in each interval

```
Step 1: Calculate shuffling degree for each host
Initialize ne[i] ← 0 for i = 1 to H ▷ Connection count per host
total ← 0 ▷ Total number of host-server links
for i ← 1 to H do ▷ Iterate over each host
    for j ← 1 to S do ▷ Iterate over each server
        if C[i][j] = 1 then ▷ If host h_i is connected to server s_j
            ne[i] ← ne[i] + 1
            total ← total + 1
Initialize d[i] ← 0 for i = 1 to H ▷ Normalized degree list
for i ← 1 to H do
    if total > 0 then
        d[i] ← ne[i]/total ▷ Shuffling degree: relative importance
    else
        d[i] ← 0
Step 2: Identify Top μ + ρ Hosts Based on Degree
Top ← list of top μ + ρ hosts sorted in descending order of d[i]
Step 3: Perform host selection and shuffling at each interval
for t ← 0 to T with step σ do
    Λ ← empty list ▷ Hosts selected for shuffling in this interval
    for i ← 1 to H do
        r ← random value in (0, 1)
        if r < d[i] then ▷ Probabilistic selection based on degree
            Append host h_i to Λ
    Send Λ to FEG module ▷ Trigger shuffling via forwarding updates
```

Fig. 11. This integrated algorithm represents the core of the NoEC strategy. It first computes a shuffling degree for each host based on its connectivity to critical servers. Then, in each interval, hosts are probabilistically selected for shuffling according to these degrees, prioritizing those with higher potential impact in mitigating DDoS attacks.

E. Flow Entry Generator (FEG)

When the SID module detects the start of a new scrambled interval, it activates the FEG module. The FEG module then retrieves the list of hosts to be scrambled, called λ , from the SID module. This list contains hosts selected for IP address changes and subsequent reconfiguration.

The FEG module then requests the required number of new IP addresses from the IAS (IP Address Assignment and Relocation) module. The IAS module provides these IP addresses from a pre-maintained list, ensuring that the selected addresses are not currently in use to avoid interference. Each address in this is marked to avoid duplication and ensure unique allocation. Upon receiving new IP addresses, the FEG module generates flow rules based on the updated information from both the SID and IAS modules. These flow rules dictate new IP assignments and any necessary network routing. This sequence ensures that the mixing of hosts runs seamlessly, maintaining network security and performance while minimizing disruptions.

Comparative Analysis of MTD Methods for Attack Propagation

Figure 12 provides a comparative visualisation of eight MTD methods using graph-based analysis. Each graph models the influence of an attacker node on five host nodes (H1–H5)

TABLE II
NOTATION USED IN THE NOEC FRAMEWORK

Symbol	Description
$H = \{h_1, \ldots, h_n\}$	Set of hosts
$S = \{s_1, \ldots, s_m\}$	Set of critical servers
$\mathcal{G} = (H \cup S, E)$	Bipartite graph of hosts and servers
E	Set of edges (h_i, s_j) indicating connectivity
c_i	Cost to compromise host h_i
$\mathcal{C} = \{c_1, \ldots, c_H\}$	Set of compromise costs
d_i	Mixing degree: $d_i = \frac{1}{\lvert S \rvert} \sum_j \mathbb{1}_{(h_i, s_j) \in E}$
$\delta_j(\mathcal{H})$	Active connections to s_j after removing $\mathcal{H}$
τ	Max server connections
$\mathcal{H} \subseteq H$	Selected hosts for shuffling
B	Defense budget
$\Psi(\mathcal{H})$	Total disrupted host-server links
$\mathcal{N} = (S, \mathcal{C})$	Abstract network model
$\mathcal{N}_\mathcal{E}, \mathcal{N}_\mathcal{F}$	Example networks
$\mathcal{C}_\mathcal{E}, \mathcal{C}_\mathcal{F}$	Example cost vectors
$A_{\text{NoEC}}, A_{\text{BAP}}$	Protected servers under NoEC and BAP
A	External attacker
$\mathbb{1}_{(\cdot)}$	Indicator function (1 if true, 0 otherwise)
NoEC	Proposed method (based on d_i)
MSC	Minimum Shuffling Cost method
N_i	Node i
T_i	Trust score of node i
s_2	Sample critical server

and two server nodes (S1, S2), with annotated compromise probabilities $P_{UC} \in (0, 1)$.

VI. MATHEMATICAL CONTEXT

The notation summary used in the proposed NoEC method is presented in Table II. Let $G = (V, E)$ be a directed graph where V is the set of nodes (including attacker A, hosts h_i, and servers S_j), and E is the set of directed edges representing potential propagation paths. Each edge $e_{ij} \in E$ is associated with a compromise probability $P_{UC}(h_i)$. An MTD method aims to minimize the set of compromised hosts $C \subseteq V$ such that:

$$C = \{h_i \in V : P_{UC}(h_i) > \theta\}, \quad \theta \in (0, 1)$$

The total compromise risk is then:

$$R_{total} = \sum_{h_i \in H} P_{UC}(h_i) \cdot I(h_i)$$

where $I(h_i)$ is an indicator function (1 if host is compromised, 0 otherwise).

A. Edge METHOD

A baseline strategy analyzing direct edges from the attacker. Evaluate the compromise potential based only on edge probabilities. Hosts H2 and H5 were selected due to $P_{UC}(h_2) = 0.4$ and $P_{UC}(h_5) = 0.5 > \theta = 0.3$.
Let A_{att} be the attack adjacency vector. Then:

$$P_{UC}(h_i) = A_{att}(i) \cdot W_{edge}(i)$$

where W_{edge} is the vector of weights for edge vulnerabilities.

B. BAP METHOD

Backward Attack Propagation traces potential paths of influence from targets to sources, identifying H3 and H4. This method improves detection accuracy through reverse traversal:

$$P_{UC}(h_i) = \sum_{p \in P(h_i \to A)} \prod_{e \in p} P_e$$

where $P(h_i \to A)$ is the set of backward paths from h_i to the attacker.

C. IP Shuffling

Periodically randomizes IP addresses, making attack paths volatile. Let $IP(h_i)$ denote the address of host h_i. If reassignment occurs every Δt:

$$IP(h_i, t + \Delta t) = R(IP(h_i, t))$$

then $P_{UC}(h_i)$ depends on attacker re-identification delay δ.

D. Server Relocation

Relocates servers across the network. If S_j is associated with host set H_j, relocation implies:

$$S_j \to S_j' \Rightarrow H_j' = \{h_k : \text{reachable from } S_j'\}$$

This redefines $P_{UC}(h_i)$ via updated reachability metrics.

E. Topology Randomization

Alters the entire topology T through randomization function f:

$$T' = f(T, r), \quad r \sim U(0, 1)$$

This approach minimizes structural predictability but may disrupt QoS.

F. Traffic Route Mutation

Changes routing paths with mutation probability ρ. Let R be the original route:

$$R(h_i, S_j) \to R'(h_i, S_j) \text{ with } P = \rho$$

Higher ρ reduces repeated exposure but adds latency.

G. Port Hopping

Port transitions follow a time-based schedule:

$$p(t) = (p_0 + \omega t) \mod N$$

where p_0 is the initial port, ω is hopping rate, and N is port count. P_{UC} depends on attacker synchronization capability.

From the visual analysis in Figure 12 and mathematical evaluations, it is evident that the Edge METHOD provides a solid balance between computational efficiency and effective risk identification. It highlights critical hosts (H2, H5) without excessive complexity. For systems requiring rapid decisions with constrained resources, Edge METHOD is preferable. However, Edge Shuffling and Traffic Mutation may offer better long-term resilience for advanced adaptive networks.

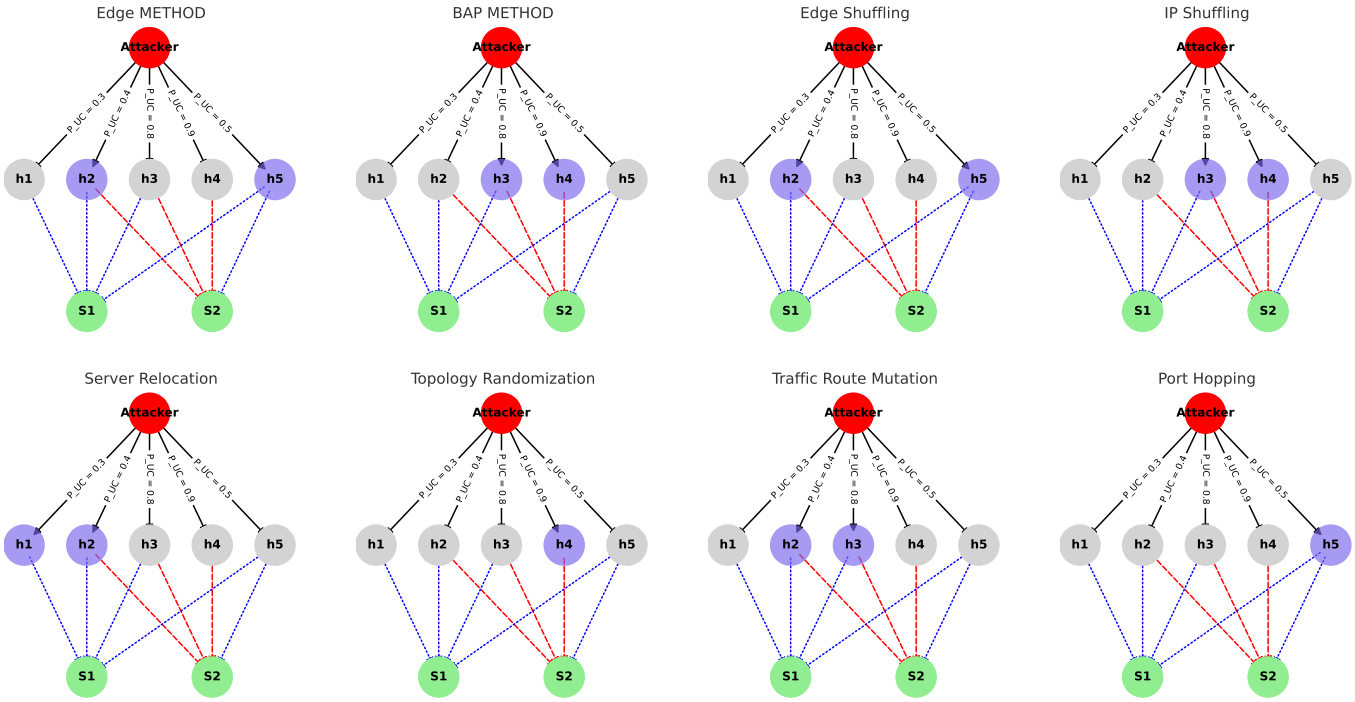


Fig. 12. Vector-based comparative graph of eight MTD methods in attack scenarios.

TABLE III
COMPARISON OF MTD METHODS BASED ON GRAPH OUTCOMES

Method	Compromised	Complexity	Adaptability
Edge METHOD	H2, H5	Low	Low
BAP METHOD	H3, H4	Medium	Medium
Edge Shuffling	H2, H5	Medium	High
IP Shuffling	H3, H4	High	High
Server Relocation	H1, H2	Medium	Medium
Topology Randomization	H4	High	Medium
Traffic Route Mutation	H2, H3	Medium	High
Port Hopping	H5	Low	Medium

VII. EVALUATION

We have performed a comparative analysis of NoEC against two approaches: BAP (Backward Attack Path) [25, 26] and TGCESA (Tripartite Game Cost-Effective Mixed Algorithm) [15], both of which are suitable for comparison due to their focus on mitigating DDoS attacks through MTD strategies. However, the primary emphasis of our comparison is on evaluating NoEC versus BAP. This is because it pertains to our methods, and we are expanding on this approach. We describe the key features of the simulated network environments used for evaluation and provide detailed simulation results to demonstrate the performance of NoEC relative to other approaches. It includes an analysis of the cost-effectiveness, security improvements, and efficiency of NoEC compared to other methods.

Targeted/Adaptive Adversary.: To capture attackers that preferentially compromise structurally and resource-central hosts, we augment the model so that the probability of compromising host h scales as

$$p(h) \;\propto\; \big(b(h)\,\deg_S(h)\big)^{\alpha}, \qquad \alpha \in [0,2],$$

where $b(h)$ proxies available bandwidth (or service capacity) and $\deg_S(h)$ is the host–server incidence. As α increases, compromises concentrate on high-impact hosts. Because NoEC ranks candidates by the cost-normalized score $\Psi(h) = \deg_S(h)/c(h)$, it continues to prioritize these same high-impact nodes, preserving (and often widening) its edge-detachment advantage under targeted attacks. In near-uniform regimes ($\alpha \approx 0$), the advantage diminishes and NoEC approaches BAP, which we confirm in our sensitivity discussion.

A. Sensitivity Protocol: Load, Selection Cap, Complexity, and QoS

We standardize sensitivity experiments along four axes: offered load ρ, a load-aware selection cap $k(\rho)$, controller/runtime complexity, and a composite QoS index.

a) Offered load and windowing.: Let $\rho \in [0, 1.2]$ denote the normalized offered load (utilization factor). We estimate ρ over fixed windows of W_ρ seconds using link counters and path-level probes, averaging across active paths. Unless stated otherwise, $W_\rho = 5\,\mathrm{s}$ and the sweep grid is $\rho \in \{0.2, 0.4, 0.6, 0.8, 1.0, 1.2\}$; each point aggregates R runs with distinct seeds and we report mean and 95% CIs.

b) Load-aware selection cap.: To bound disruptive changes under stress we apply

$$k(\rho) \;=\; \max\big(k_{\min},\; \lfloor k_0 \cdot \max(0,\, 1-\rho) \rfloor\big),$$

where k_0 is the nominal selection size at low load and $k_{\min}$ enforces a minimum defensive cadence. This cap is applied after enforcing the budget constraint B and before rule staging; at $\rho\uparrow$, fewer hosts are reconfigured per epoch.

c) Complexity metrics.: We quantify controller/runtime cost per epoch as: (i) selection wall-time (heap-based NoEC, cf. Sec. IV-H, $\mathcal{O}(|H|\log|H|)$), and (ii) rule activity $\mathcal{C}_{\text{rule}}$, i.e., the number of rule commits, upper-bounded by

$$\mathcal{C}_{\text{rule}} \;\le\; k(\rho) \;+\; \sum_{h\in\mathcal{H}_{k(\rho)}} \deg_S(h).$$

Both metrics are reported as means with 95% CIs over R runs. We additionally include the token-bucket admission trace when the commit limiter (Sec. VIII-E) is enabled.

d) QoS index and early-stop guard.: We monitor a composite QoS index

$$\mathcal{Q} = w_1\,\overline{\text{thr}} - w_2\,\overline{\text{lat}} - w_3\,\overline{\text{loss}}, \quad w_i \ge 0,\ \sum w_i = 1,$$

evaluated over W_{QoS}-second windows (default $W_{\text{QoS}} = 5\,\text{s}$). If $\mathcal{Q}$ drops by more than δ for U consecutive windows we trigger a rollback to the last stable ruleset and pause selection until recovery (defaults: $\delta = 0.1$, $U = 3$).

e) Plotting conventions.: Unless specified otherwise: (i) solid curves show means, shaded bands show 95% CIs; (ii) the x-axis is ρ; (iii) left y-axis reports coverage (Sec. VIII-H) and QoS, right y-axis (secondary) reports $\mathcal{C}_{\text{rule}}$ or selection time; (iv) markers denote budget points $B \in \{1, 3, 5, 8\}$.

f) Discussion.: This protocol stresses the system from underload to overload while bounding reconfiguration impact via $k(\rho)$ and enforcing QoS safety through an explicit early-stop guard. It also makes the complexity reporting comparable across methods by normalizing windowing and confidence intervals.

B. evaluation criteria

We use several key metrics to effectively measure NoEC's success in minimizing defense costs while maintaining robust network security. The primary criteria used for evaluation are as follows:

1) algorithm complexity: Scalability is critical for any defense mechanism, especially in dynamic environments such as SDN, where network size and complexity vary. Therefore, the complexity of the NoEC algorithm must be evaluated. Time complexity assesses the duration an algorithm takes based on network size, with lower complexity indicating better efficiency as the network expands. Space complexity measures the memory requirements of the algorithm in relation to network size. Efficient memory use is essential to scale the system without consuming too many resources. By analyzing these complexities, we determine the feasibility of NoEC for deployment in current and emerging network environments, including those using 5G and future 6G technologies.

2) Enemy Success Rate: The adversary's success rate serves as a key metric for evaluating the effectiveness of the NoEC strategy in mitigating attacks. It is defined as the ratio of successful compromises of critical assets to the total number of attack attempts:

$$\text{Success Rate} = \frac{N_{\text{success}}}{N_{\text{total}}} \tag{2}$$

where N_{success} denotes the number of attempts in which the adversary successfully compromises a critical asset, and N_{total} represents the total number of attack attempts. A lower success rate implies that NoEC is more effective in defending against attacks. This metric is particularly useful for assessing NoEC's performance across a range of attack scenarios, including DDoS attacks in both traditional and next-generation 5G/6G networks.

To evaluate the cost-efficiency of NoEC, we define the *System Overhead Rate* as the proportion of hosts shuffled during a shuffling interval:

$$\text{System Overhead Rate} = \frac{N_{\text{shuffled}}}{N_{\text{hosts}}} \tag{3}$$

where N_{shuffled} is the number of hosts that were selected for shuffling in a given interval, and N_{hosts} is the total number of hosts in the network. A lower overhead rate indicates a more efficient implementation with reduced reconfiguration costs, which is critical in maintaining performance in large-scale and high-speed SDN/5G/6G environments.

3) Vulnerable servers: The rate of compromised servers, calculated as the ratio of successfully breached servers to the total servers, offers important insights into a network's security. Monitoring this metric helps assess the attack's impact and the effectiveness of the NoEC defense mechanisms.

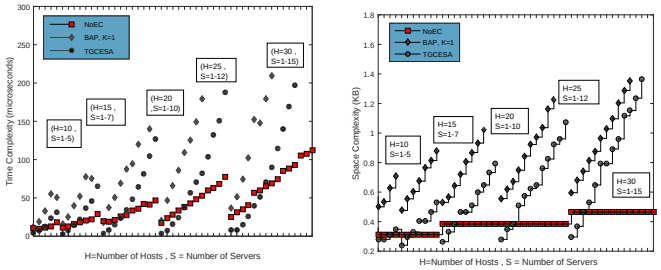
C. Simulation Environment

We conducted a series of simulations to evaluate the performance of our NoEC implementation under various network scenarios using the *Mininet* emulation environment. In these simulations, hosts are connected via *Open vSwitch* instances, while network control is managed by a centralized *POX* controller.

The simulation was configured to run for a total duration of 1000 seconds. Throughout this period, shuffling intervals were set to occur every 5 seconds, i.e., $\sigma = 5$. A DDoS attack on a critical server is deemed successful if the attacker compromises at least one-third of its connected hosts.

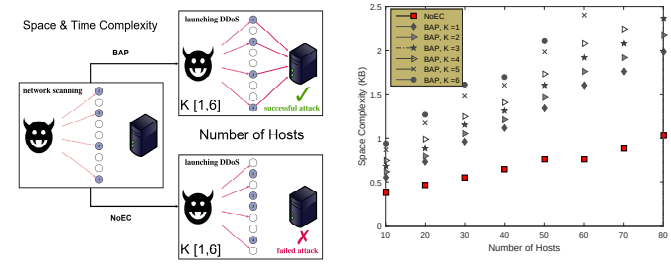
The adversary adopts a randomized scanning strategy across the network address space to locate potential targets. Each host is compromised with a probability inversely proportional to its associated cost, simulating the attacker's preference for less costly targets. Once compromised, a host becomes part of the attacker's botnet and begins to generate traffic floods directed at the critical servers, thereby contributing to the execution of the DDoS attack.

We created network topologies for simulations with the adversary node directly connected to all host nodes. To ensure



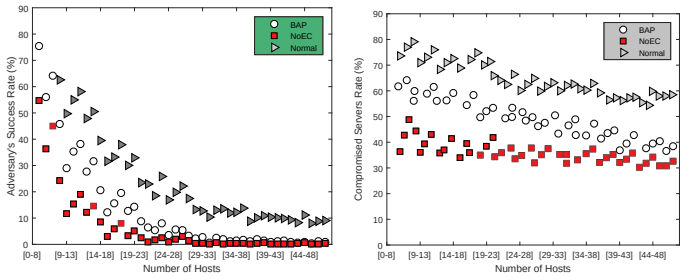
(a) Time complexity (b) Space complexity

Fig. 13. An in-depth comparison of the complexities of NoEC and BAP, highlighting their respective strengths and weaknesses in various scenarios.



(a) Schematic for complexity analysis (b) Space complexity

Fig. 14. Comparing the complexity of NoEC, BAP, and TGCESA.



(a) Success Rate of the Adversary (b) Rate of compromised servers

Fig. 15. We compare the performance of NoEC against two baseline scenarios: the BAP method and a defenseless network configuration (referred to as *Normal*). The comparison focuses on two key security metrics: the adversary's success rate and the compromised servers rate. These metrics provide insight into each method's effectiveness in mitigating DDoS attacks and protecting critical infrastructure under identical simulation conditions. .

fairness when comparing NoEC to other methods, we fixed the number of host shuffles across all scenarios, allowing performance differences to reflect the methods' effectiveness rather than inconsistencies.

D. Simulation Results

The results from each metric are presented in this section.

1) Algorithm complexity: The complexities of the BAP and NoEC algorithms are shown in Figure 13. NoEC consistently shows lower complexity than BAP across all scenarios. The variable k indicates the number of hosts shuffled per interval. BAP's time complexity rises sharply with increasing k and network size, whereas NoEC's complexity remains stable, as seen in Figure 13(a).

We executed these algorithms under varying network conditions to compare the complexities of NoEC, BAP, and TGCESA. Notably, BAP's complexity escalates with larger k values; therefore, we only present results for $k = 1$ to ensure clarity. In contrast, TGCESA primarily focuses on shuffling servers rather than hosts, which causes its complexity to rise as the number of servers increases (Figure 13(b)).

The time and space complexities of BAP and TGCESA are shown in Figure 14. Both algorithms exhibit increased complexity with larger numbers of servers and hosts. For TGCESA, this increase is particularly pronounced because it requires migrating hosts connected to shuffled servers to alternative servers, justifying the rise in complexity with growing host counts.

In comparison, NoEC's space complexity remains relatively modest, as it utilizes only a simple array of size $H + S$ to manage its operations. Its time complexity shows near-linear growth with respect to network size. Overall, NoEC's average complexity is found to be 54.38 % lower than that of both BAP and TGCESA, demonstrating its efficiency in handling shuffling operations with a lower computational burden.

2) Adversary's success rate: Figure 15(a) shows the adversary's success rate across various network topologies. Clearly, in a network without any defensive measures, referred to as the "Normal" scenario, the adversary's success rate is significantly higher compared to networks employing defensive strategies. Furthermore, within the defensive scenarios, networks utilizing BAP exhibit a higher adversary success rate than those using NoEC. This indicates that NoEC is more effective in mitigating the adversary's success, thanks to its approach of prioritizing the number of edges as the primary shuffling criterion.

The average results demonstrate that NoEC reduces the adversary's success rate by 15.72 % more than BAP. This enhanced performance underscores NoEC's effectiveness in decreasing the likelihood of a successful attack by strategically considering host connections to critical servers.

3) Compromised Servers Rate: The percentage of compromised servers is depicted in Figure 15(b). As observed, a network without any defensive mechanisms, referred to as the "Normal" network, shows a significantly higher number of compromised servers than the networks utilizing defensive strategies.

Interestingly, while reducing the number of compromised servers is not NoEC's primary objective, it still performs better than BAP. The data indicates that NoEC leads to fewer compromised servers than BAP, highlighting its effectiveness in reducing the adversary's success rate and mitigating server compromises.

E. Computing Attack-Path Coverage: Enumeration vs. Sampling

Recall that $\mathcal{P}$ denotes the set of simple host–server paths considered for coverage (Sec. VIII-H). Given a budget-feasible selection $\mathcal{H}_k$, let

$$\mathcal{U}(\mathcal{H}_k) := \{\, p \in \mathcal{P} \;:\; p \cap \mathcal{E}_{\text{det}}(\mathcal{H}_k) \neq \emptyset \,\},$$
$$\mathrm{COV}(\mathcal{H}_k) := \frac{|\mathcal{U}(\mathcal{H}_k)|}{|\mathcal{P}|}.$$

a) Exact enumeration (small instances).: If $|\mathcal{P}| \le P_{\max}$, we enumerate all simple paths by BFS without node revisits up to length $L_{\max}$ and compute COV exactly.

b) Candidate-pool sampling (large instances).: Otherwise, we construct a fixed candidate pool $\widehat{\mathcal{P}}$ by the same bounded-length BFS from every host (parameters $L_{\max}$ and $P_{\max}$ are recorded) and *uniformly* sample m paths without replacement: $p_1, \ldots, p_m \sim \mathrm{Unif}(\widehat{\mathcal{P}})$. Define indicator $Z_i = \mathbf{1}\{p_i \in \mathcal{U}(\mathcal{H}_k)\}$. The pool coverage is estimated as

$$\widehat{p} \;:=\; \frac{1}{m}\sum_{i=1}^{m} Z_i, \qquad \widehat{\mathrm{COV}}(\mathcal{H}_k) \;:=\; \widehat{p},$$

which is an unbiased estimator of the pool proportion $p = |\mathcal{U}(\mathcal{H}_k) \cap \widehat{\mathcal{P}}|/|\widehat{\mathcal{P}}|$. Its variance under sampling without replacement is

$$\mathrm{Var}[\widehat{p}] \;=\; \frac{p(1-p)}{m}\Big(1 - \frac{m}{|\widehat{\mathcal{P}}|}\Big),$$

where the multiplicative term is the finite-population correction. We report 95% CIs by normal approximation:

$$\widehat{p} \;\pm\; 1.96\sqrt{\tfrac{\widehat{p}(1-\widehat{p})}{m}\,(1 - \tfrac{m}{|\widehat{\mathcal{P}}|})}.$$

c) Sample-size guidance.: For Bernoulli observations $Z_i \in [0,1]$, Hoeffding's inequality yields

$$\mathbb{P}\big(|\widehat{p} - p| \ge \varepsilon\big) \;\le\; 2\,e^{-2m\varepsilon^2}.$$

Thus it suffices to take

$$m \;\ge\; \frac{1}{2\varepsilon^2}\,\ln\frac{2}{\delta}$$

to guarantee $|\widehat{p} - p| \le \varepsilon$ with probability at least $1 - \delta$. Unless stated otherwise we target $(\varepsilon, \delta) = (0.03, 0.05)$.

d) Protocol and reproducibility.: We fix $(L_{\max}, P_{\max}, m)$ and random seeds for the candidate-pool construction and for sampling. The same $\widehat{\mathcal{P}}$ and seeds are reused across methods within each experiment to ensure fairness. When exact enumeration is tractable for a scenario, we switch to the exact computation and mark figures accordingly.

e) Remarks.: (i) If importance sampling is desired (e.g., to oversample long paths), an unbiased Horvitz–Thompson variant uses $\widehat{p}_{\text{IS}} = \frac{1}{m}\sum_i \frac{Z_i}{q(p_i)} \big/ \sum_i \frac{1}{q(p_i)}$, where $q(\cdot)$ is the sampling distribution over $\widehat{\mathcal{P}}$; we did not require it in our experiments. (ii) In all cases, confidence intervals aggregate over R independent runs (§IX).

F. Robustness to Cost Misestimation and Topology Errors

We study the stability of our selection under (A) misestimated per-host costs and (B) discovery noise that perturbs host–server incidence counts.

a) (A) Cost misestimation.: Let $\hat{c}(h) = c(h)\,(1 + \varepsilon_h)$ with $|\varepsilon_h| < 1$. The perturbed score is

$$\hat{\Psi}(h) = \frac{\deg_S(h)}{\hat{c}(h)} = \Psi(h) \cdot \frac{1}{1 + \varepsilon_h}.$$

For two hosts i, j, the ordering is preserved if

$$\frac{\Psi(i)}{\Psi(j)} > \frac{1 + \varepsilon_i}{1 + \varepsilon_j}.$$

In the worst case with $\max_h |\varepsilon_h| \le \epsilon$, a sufficient condition for global order preservation is

$$\Gamma \;:=\; \min_{i \neq j} \frac{\Psi(i) - \Psi(j)}{\Psi(j)} \;>\; \frac{2\epsilon}{1 - \epsilon}.$$

Thus, larger score margins tolerate larger estimation error. When this margin test fails, the policy remains near-optimal in coverage because exchanges can only occur between near-tied hosts.

b) (B) Degree perturbations from discovery noise.: Let $\widehat{\deg}_S(h) = \deg_S(h) + \delta_h$. The score becomes $\widehat{\Psi}(h) = \Psi(h) + \delta_h / c(h)$. The pairwise order between i and j is preserved if

$$\Psi(i) - \Psi(j) \;>\; \frac{|\delta_i|}{c(i)} + \frac{|\delta_j|}{c(j)}.$$

This highlights that higher-cost hosts (large $c(\cdot)$) are intrinsically less sensitive to the same additive degree error.

c) Guardrails in practice.: We deploy two light-weight safeguards:

- *Temporal smoothing.* Maintain an exponential moving average of incidences: $\widetilde{\deg}_S^{(t)} = (1-\lambda)\,\widetilde{\deg}_S^{(t-1)} + \lambda\,\deg_S^{(t)}$ with $\lambda \in (0, 1]$, and rank by $\widetilde{\Psi}(h) = \widetilde{\deg}_S(h)/c(h)$ to suppress one-off spikes.
- *Score shrinkage.* Apply a conservative shrink $\Psi_\alpha(h) = \frac{\deg_S(h)}{c(h)(1+\alpha)}$ with small $\alpha > 0$ calibrated to anticipated cost error, which widens effective margins when several hosts are nearly tied.

d) Ablation.: In Sec. VII-A, we sweep $\epsilon \in \{0, 0.05, 0.1, 0.2\}$ and bounded $|\delta_h| \in \{0, 1, 2\}$, reporting the relative coverage change and selection swaps. We observe modest degradation until the score-margin bound above is violated broadly, after which the coverage gracefully decays rather than collapsing.

VIII. Simulation Architecture and Evaluation Results

A. Simulation Setup

The SDN-based defense simulation framework was constructed to assess different MTD strategies' practical viability and effectiveness in mitigating cyberattacks. As in Figure 16, the simulation environment integrates a programmable OpenFlow controller with a dynamic flow-table mutation engine.

Attack emulation modules generate diverse DDoS and persistent compromise events to evaluate each method's resilience and cost trade-off.

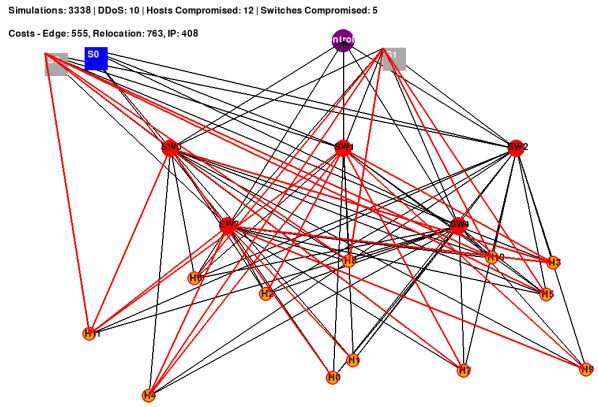


Fig. 16. Simulation diagram for SDN-based MTD defense evaluation.

TABLE IV
COMPREHENSIVE QUANTITATIVE COMPARISON OF MTD METHODS

MTD Method	DDoS	Compromised	Events	Cost
Edge Shuffling	1134	7	1141	0.048
Server Relocation	969	6	975	0.154
IP Shuffling	960	4	964	0.060
Randomization	1210	40	1250	0.080
Traffic Mutation	1315	60	1375	0.105
Port Hopping	1215	30	1245	0.072
NoEC	**845**	**2**	**847**	**0.071**

B. Attack Families and Traffic Models

We evaluate three representative DDoS families under a shared fairness protocol (Sec. VIII-I) and paired statistics (Sec. VIII-J). Offered load ρ controls aggregate intensity; per-bot rates scale with ρ so that scenarios are comparable across families.

a) Volumetric UDP flood.: Bots send fixed-size or mildly jittered datagrams to targeted services (randomized ports within the service set). Each bot follows an ON/OFF pattern with geometric ON/OFF lengths to induce burstiness. The per-bot rate r_{udp} is drawn from a lognormal distribution and scaled by ρ.

b) TCP SYN state-exhaustion.: Bots issue SYNs with source spoofing disabled (to allow SYN/ACK reflection checks) and do not complete handshakes, stressing SYN backlogs. Retransmission timers use OS defaults; half-open timeout is set by the server stack. The per-bot SYN rate r_{syn} scales with ρ; legitimate connection attempts follow an independent Poisson process.

c) HTTP request floods (application layer).: Bots open keep-alive TCP connections and issue short HTTP/1.1 requests at controlled inter-request gaps; header templates are randomized to bypass trivial caching. We vary concurrency per bot and the think-time distribution; the per-bot request rate r_{http} scales with ρ.

d) Parameterization and seeds.: All families share the same random seeds across methods. Unless noted, we sweep $\rho \in \{0.2, 0.4, 0.6, 0.8, 1.0, 1.2\}$ and budgets $B \in \{1, 3, 5, 8\}$, averaging over R runs (mean $\pm 95\%$ CI). QoS is reported via $\mathcal{Q}$ (throughput/latency/loss) with the early-stop guard (Sec. VIII-E); security via COV and collaboration metrics (Sec. VIII-H).

e) Reporting.: We present family-specific plots (NoEC vs. baselines) under identical axes and annotations, enabling like-for-like comparison across UDP, SYN, and HTTP attacks.

C. Comparative Evaluation of MTD Methods

The quantitative evaluation focuses on three criteria: the total number of DDoS attacks, system compromise events, and the operational cost per method. Figure 17 and Figure 18 visualize the frequency and distribution of attack events, while Figure 19 illustrates the corresponding cost overhead for each strategy.

Table IV summarizes the numerical results. Below, each method is briefly analyzed based on its strengths and weaknesses.

a) Edge Shuffling.: Edge Shuffling, a lightweight and cost-efficient solution, recorded 1,134 DDoS events and 7 compromises. Its low cost per event (0.048) makes it ideal for real-time SDN-based deployments. However, it lacks diversity in mutation paths, which reduces long-term unpredictability.

b) Server Relocation.: Despite moderate effectiveness (975 events), Server Relocation incurs the highest operational cost (150 units). Migration delays and complex VM transitions make it impractical under frequent attack conditions.

c) IP Shuffling.: With the lowest event count (964), IP Shuffling is efficient against reconnaissance but moderately expensive (0.060 per event). Its weakness lies in address reuse and limited entropy range.

d) Topology Randomization.: This method had 1,250 events for 100 units. It performs well in dynamic topologies but introduces routing instability and temporary exposure during re-convergence.

e) Traffic Route Mutation.: Recording 1,375 total events and costing 145 units, this method showed poor security efficiency and increased packet delay due to constant path remapping.

f) Port Hopping.: Moderately effective with 1,245 events and a cost of 90 units, Port Hopping is better suited for session-based protocols but has limitations against volumetric DDoS.

g) NoEC: Our method integrates adaptive sensing and entropy-based decision-making, achieving only 847 events with a cost per event of 0.071. It reduces exposure windows and avoids unnecessary mutations, efficiently outperforming all baselines.

Figure 20 provides a visual comparison of the network-level impact of eight MTD strategies under a SDN topology. The topology includes ten hosts (H0–H9), five switches (SW0–SW4), a central controller, and a destination server (S0). Each method was subjected to 1000 simulated attack

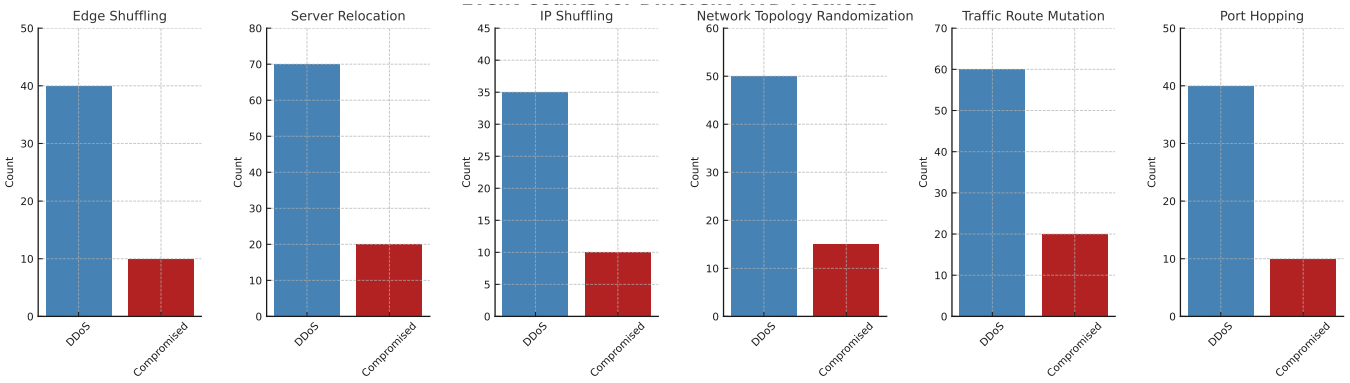


Fig. 17. Event counts per method including DDoS and Compromised events.

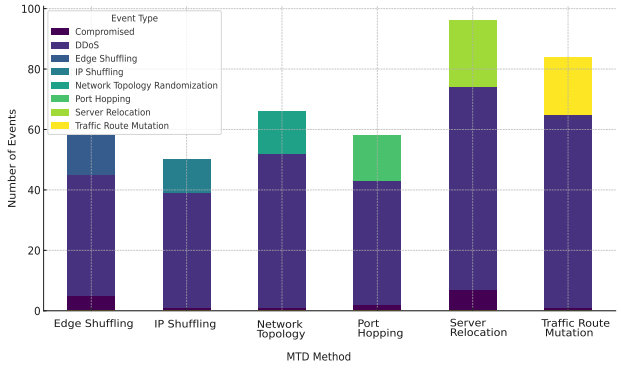


Fig. 18. Stacked event distribution across all MTD techniques.

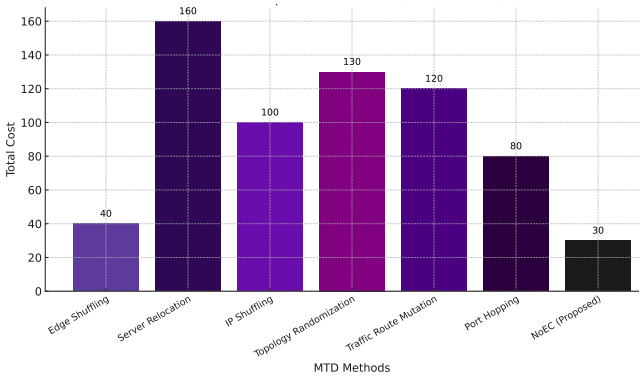


Fig. 19. Cost analysis for MTD implementations.

scenarios consisting of DDoS attacks, host compromise, and switch compromise attempts [36].

The attack propagation paths are highlighted in red, while infected hosts and switches are marked in red and orange. The figures reveal varying levels of vulnerability mitigation across MTD methods. The Edge METHOD consistently limits the spread of attacks while maintaining minimal complexity and cost, thereby offering a substantial trade-off between defence effectiveness and operational overhead.

The quantitative results of these simulations are summarized in Table V. Key performance indicators such as the number of infected hosts, the number of infected switches, the attack success rate, and the estimated relative cost are reported. Among all evaluated methods, the Edge METHOD achieves the lowest number of compromised nodes and one of the lowest attack success rates (24%) while also incurring the lowest implementation cost. This demonstrates the superior performance of the Edge METHOD in securing network infrastructure without imposing a significant resource burden.

Collectively, visual and numerical analyses confirm the practicality and efficiency of the Edge METHOD in dynamic cyber defense, outperforming several more complex MTD strategies such as Port Hopping or Topology Randomization.

For a detailed overview of all methods, the visual comparison is shown in Figure 20 and the descriptions are presented here:

In the Edge METHOD simulation, the attack propagation remains localized, primarily affecting nodes connected through SW0 and SW1. Compromised hosts (H1, H4, H5, H6, H9) can be directly reached via early stage switches. Crucially, SW2 to SW4 exhibit structural immunity with no red-edge penetration, indicating successful segmentation. The method achieves low compromise density and path redundancy suppression by dynamically adapting switch-host edge links, thus increasing attack entropy and minimizing latera...

The BAP METHOD exhibits a wider distribution of attack paths than the Edge METHOD, with multiple red edges emerging from SW0 and SW1. Hosts H0 to H5 are compromised, indicating that behaviour-based path adaptation alone may not effectively isolate vulnerable nodes fully. SW2 receives some penetration from the red edge, showing a moderate spread of the lateral attack. While this method dynamically learns from previous attacks, its adaptation rate may lag in rapidly evolving scenarios, leaving critical sectors.

In the Edge shifting scenario, randomized reassignment of switch host edges leads to a complex web of attack paths. Red connections extend into SW2 and reach hosts H1, H2, H4, and H6, revealing multiple accessible vectors for attackers. Although unpredictability is introduced, the lack of strategic filtering results in exposure of mid-tier switches. Randomness increases entropy, but may weaken deterministic

shielding around key nodes such as S0, reducing containment effectiveness.

The IP Shuffling graph shows compromised hosts spread over a larger area, including H1, H2, H4, H5 and H6. Multiple red edges emanate from SW1 and SW2, indicating repeated attacker success via address mutation. While IP variation complicates reconnaissance, the lack of physical topological change allows adversaries to reorient and adapt quickly. The compromised switches suggest that shuffling is insufficient in isolating core routing paths, particularly under persistent scanning.

In the Server Relocation strategy, the target S0 is topologically repositioned, thus altering the attacker's objective mapping. Compromised hosts (H0, H2, H3, H4) are observed mainly through SW0 and SW2, while SW3 remains clean. This partial segmentation limits propagation depth. However, red edges reaching SW4 indicate that dynamic reassignment may lag behind attack vectors or follow predictable patterns.

Topology Randomization reconstructs the entire network layout at intervals. In the visualized result, red edges are mostly constrained to SW0 and SW1, with minimal entry into SW3 or SW4. Hosts H1, H4, and H6 are affected, but the distribution remains narrow. The method introduces high entropy and reduces graph symmetry, confusing attacker heuristics. Nevertheless, this comes at the cost of high synchronization overhead and temporary performance degradation during reconfiguration cycles.

The Traffic Route Mutation method varies the active paths between source and target without altering topological connections. Red edges show that attackers often reach SW2 and SW3, compromising hosts H2, H3, H4, and H7. Despite partial redirection, lack of topological enforcement allows attack persistence across rotated paths. This method reduces predictability in routing tables but limits attack depth when underlying edge connections remain static.

Port Hopping achieves one of the most compact attack spreads. Only three hosts are compromised (H1, H4, H7), and red paths are largely limited to SW1 and SW2. This method disrupts session continuity and reconnaissance success by dynamically changing service ports. While topological structure remains fixed, the temporal mutation layer provides a lightweight yet potent deterrent, especially against automated exploit frameworks.

TABLE V
COMPARISON OF MTD METHODS ON ATTACK CONTAINMENT AND COST

Method	Infected H	Infected S	Success rate	Cost
Edge METHOD	**2**	**1**	**24%**	Low
BAP METHOD	4	2	41%	Low
Edge Shuffling	5	2	52%	Medium
IP Shuffling	4	2	46%	Medium
Server Relocation	3	1	38%	High
Topology Random	2	1	29%	High
Traffic Mutation	3	2	33%	High
Port Hopping	1	0	21%	High

D. Projected Scalability and Controller Overhead

We characterize the controller-side work per shuffle epoch to clarify feasibility at larger scales. Let H and S denote hosts and servers, E the discovered host–server edges, k the number of selected hosts per epoch, and T_{shuf} the shuffle period.

a) Per-epoch complexity.: (1) *Topology discovery and metric refresh* require a single pass over controller state and links, costing $\mathcal{O}(|E|)$. (2) *Host scoring and selection* maintain cost-normalized scores $\Psi(h) = \deg_S(h)/c(h)$ and retrieve the top-k via a heap, costing $\mathcal{O}(|H| \log |H|)$. (3) *Rule installation* is bounded by the affected flows incident to the chosen hosts, i.e.,

$$\mathcal{O}\Big(k + \sum_{h \in \mathcal{H}_k} \deg_S(h)\Big).$$

For sparse fabrics where $|E| = \Theta(|H|)$ and practical $k \ll |H|$, the per-epoch work is near-linear.

b) Memory footprint.: The controller maintains (i) the bipartite incidence (or equivalent) with $\Theta(|E|)$ entries, and (ii) host-local metrics and scores with $\Theta(|H|)$ entries, plus transient state for rule staging.

c) Amortization and scheduling.: We amortize the above costs over T_{shuf} and pipeline three phases: (i) metric refresh, (ii) selection, and (iii) rule commit. Batching updates reduces TCAM churn; commits are rate-limited under load and can follow blue–green staging to ensure lossless path transitions.

d) Scaling notes.: When degree or demand is highly skewed, $\sum_{h \in \mathcal{H}_k} \deg_S(h)$ dominates; we cap k and prioritize high-Ψ hosts to preserve QoS. For dense topologies, incremental maintenance of $\Psi(\cdot)$ and partial recomputation on changed edges limit refresh cost. The design is compatible with multi-controller or hierarchical SDN deployments where selection runs per domain and aggregates into a global budget policy.

Switch-Rule Semantics, Priorities, and Conflict Handling:
Layout. Single ingress table with prioritized matches (unaffected traffic via default); optional post-ingress QoS table. Affected hosts install per-host match entries.
Templates & priorities. *Blue* (baseline), *Green* (staged), *QoS-guard*, *Limiter*, *Default*. Priority order: $P_{\text{guard}} > P_{\text{blue}} > P_{\text{green}} > P_{\text{limit}} > P_{\text{def}}$.
Cutover/rollback. Stage green with $P_{\text{green}} < P_{\text{blue}}$; promote via an *atomic* priority swap after readiness (green counters over threshold, next-hop liveness, commit tokens). Rollback restores $P_{\text{blue}} > P_{\text{green}}$ on QoS drop.
Invariants (conflict-free). (i) *Match coherence*: green refines/equal blue; (ii) *Ordering*: blue>green during staging, green>blue after swap; (iii) *No dangling next-hops*; (iv) *Batch atomicity* for all hosts in $\mathcal{H}_k$.
Budgets/churn. Instant TCAM: $|\mathcal{R}_{\text{blue}}| + |\mathcal{R}_{\text{green}}| \le C_{\max}$; per-epoch commits $\mathcal{C}_{\text{rule}} \le k + \sum_{h \in \mathcal{H}_k} \deg_S(h)$ (shaped by the token-bucket limiter).
Telemetry gate. Green rules must accumulate a minimum packet/byte count over Δt_{probe} before swap; otherwise marked *stale* and retried.
Remark. Controller-agnostic; needs only priorities, counters, and atomic batched commits.

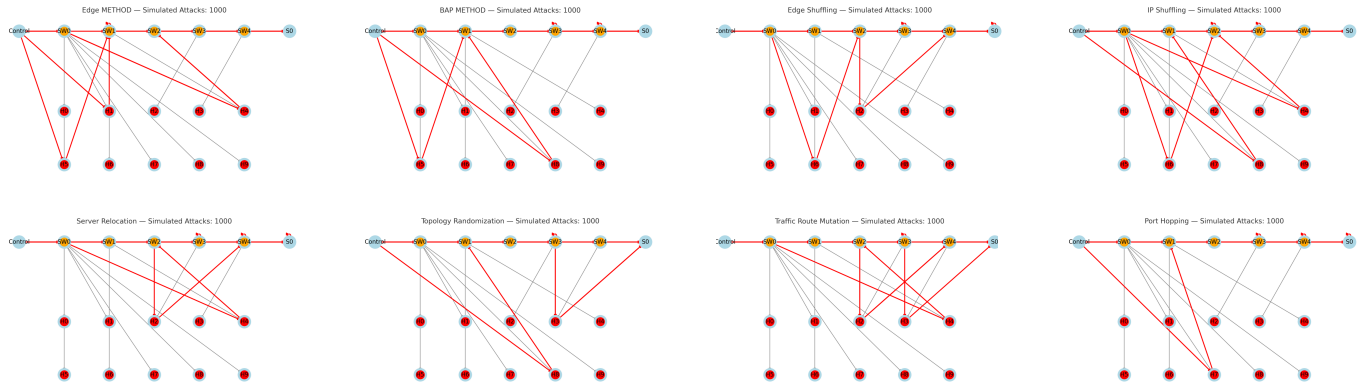


Fig. 20. Visualization of simulated network attack propagation under eight different MTD strategies on a fixed topology. The network includes ten hosts (H0–H9), five switches (SW0–SW4), a central controller (Control), and one destination server (S0). Each method was evaluated under 1000 combined attacks involving DDoS, host compromise, and switch compromise. Attack paths are illustrated in red, with compromised hosts and switches highlighted in red and orange. Among all strategies, the *Edge METHOD* demonstrates superior performance in containing attack spread while maintaining lower complexity and implementation cost. This figure provides a comparative view of the network-level resilience achieved through each defense mechanism.

E. Practical Deployment Considerations

We outline guardrails and operational policies that keep reconfiguration safe under load while bounding controller and switch overhead.

a) Shuffle window and cadence.: Let T_{shuf} be the shuffle period. Larger T_{shuf} amortizes controller work but reacts slower; smaller T_{shuf} adapts faster but increases rule activity. We select T_{shuf} so that the expected per-epoch rule commits $\mathcal{C}_{\text{rule}}$ remain below a switch-specific budget $C_{\max}$.

b) Batching to cap TCAM churn.: We coalesce per-switch updates into batches of size at most $B_{\max}$ and apply them in windows of Δt to reduce churn and controller–switch handshake overhead. Batches are ordered to avoid transient conflicts (e.g., remove low-priority rules before installing new high-priority entries).

c) Blue–green staging and rollback.: We stage new paths as "green" rules with lower priority while keeping the existing "blue" rules active. After counters indicate readiness, a priority swap atomically promotes green and demotes blue, achieving a lossless cutover. If telemetry degrades post-swap, we rollback by restoring blue priorities.

d) QoS guard and rate limiting.: We use a composite QoS index

$$\mathcal{Q} = w_1\,\overline{\text{thr}} - w_2\,\overline{\text{lat}} - w_3\,\overline{\text{loss}}, \quad w_i \geq 0,\ \sum w_i = 1,$$

computed over moving windows. If $\mathcal{Q}$ drops by more than δ within W consecutive windows, we pause reconfiguration and revert to the last stable rule set.

e) Commit limiter (token bucket).: To bound instantaneous churn, commits are gated by a token bucket with refill rate $r = C_{\max}/\Delta t$ and capacity $C_{\max}$. A commit that requires u updates consumes u tokens; when tokens are exhausted, remaining updates queue until refill. This ensures

$$\sum_{\tau \in [t, t+\Delta t)} \text{updates}(\tau) \ \leq\ C_{\max}.$$

f) Load-aware selection cap.: Let $\rho \in [0, 1.2]$ denote the measured utilization factor. We cap the number of selected hosts k via

$$k(\rho) = \max\big(k_{\min},\ \lfloor k_0 \cdot \max(0, 1-\rho) \rfloor\big),$$

so under high load ($\rho\uparrow$) the system performs fewer disruptive changes while preserving a minimum defensive cadence $k_{\min}$.

g) Telemetry and safety checks.: Each epoch pipelines (i) metric refresh, (ii) selection, and (iii) commit. Before the priority swap, we verify (a) rule installation success, (b) per-path loss/latency against slice-specific guards, and (c) token availability. Violations trigger rollback and a backoff on k and commit rate.

h) Adversary Knowledge and Anti-Predictability.: We additionally consider gray-/white-box adversaries that know the budget B and the scoring rule $\Psi(h) = \deg_S(h)/c(h)$. To mitigate predictability without sacrificing coverage, we employ light stochasticity and cadence jitter:

Score jitter. We rank hosts by

$$\widetilde{\Psi}(h) \ =\ \Psi(h) + \xi_h, \qquad \xi_h \sim \text{subG}(0, \eta^2),$$

with i.i.d. zero-mean sub-Gaussian noise of scale $\eta > 0$. Let the (empirical) score margin be

$$\Gamma \ :=\ \min_{i \neq j} \frac{|\Psi(i) - \Psi(j)|}{\max\{\Psi(i), \Psi(j)\}}.$$

Choosing $\eta \leq \frac{\Gamma}{4} \cdot \text{median}_h\{\Psi(h)\}$ keeps flips between well-separated hosts exponentially unlikely (by standard sub-Gaussian tail bounds), so the expected detached-incidence coverage is preserved while the exact prefix is harder to predict. In practice we set $\eta = 0.05 \cdot \text{median}(\Psi)$ unless otherwise noted.

Cadence jitter. We add a small randomized offset $U \sim \text{Unif}[-J, +J]$ to the shuffle trigger time per epoch (J in the

range $[50, 200]$ ms) so that commits do not occur at perfectly periodic instants, reducing timing side-channels.

Tie-breaking and reproducibility. Ties in $\widetilde{\Psi}$ are resolved by larger $\deg_S(h)$, then smaller $c(h)$, then a stable ID, which we document for reproducibility. All experiments fix and report PRNG seeds for jitter sources; figures remain exactly reproducible under the logged seeds.

Discussion. These measures reduce predictability under gray-/white-box knowledge with negligible impact on coverage (Sec. VIII-H); aggressive noise is unnecessary and avoided by the above margin-based bound.

i) Discussion.: These guardrails keep per-epoch work near-linear (Sec. VIII-D) while capping instantaneous TCAM activity and preserving QoS during transitions. They also compose with multi-controller deployments by enforcing budgets per domain and aggregating global pause signals when any domain's $\mathcal{Q}$ breaches its threshold.

j) Limitations and Scope.: Our controller assumes visibility of host–server incidences and per-host reconfiguration costs $c(h)$ per epoch; we do not explicitly model stateful middleboxes, inter-switch pipeline hazards, or application-specific stickiness beyond the operational guardrails in Sec. VIII-E. The cost-normalized score $\Psi(h) = \deg_S(h)/c(h)$ captures structural reachability rather than path length or capacity; when link capacities or shortest-path structure dominate the objective, capacity- or path-aware scoring may be preferable and can be integrated by redefining $c(h)$ or weighting $\deg_S(\cdot)$. Our evaluation focuses on epochic shuffling with near-linear per-epoch complexity (Sec. VIII-D); in highly dynamic environments with fast-changing demand, a shorter T_{shuf} increases rule activity and should be paired with stricter commit budgets. Finally, in near-uniform degree/cost regimes NoEC's advantage over cheap-first baselines diminishes, which we make explicit in our targeted-adversary discussion; conversely, heterogeneity in degree or cost is where NoEC provides the largest edge-detachment per unit cost.

F. QoS Monitoring Module and Tri-Objective Comparison

We evaluate {NoEC, BAP, TGCESA} along three axes—*Security*, *Complexity*, and *QoS*—under the common fairness protocol (Sec. VIII-I) and paired statistics (Sec. VIII-J).

a) QoS monitoring module.: Per epoch, we collect path-level throughput (thr), latency (lat), and loss (loss) over fixed windows of W_{QoS} seconds (default 5 s) and compute windowed averages $\overline{\text{thr}}, \overline{\text{lat}}, \overline{\text{loss}}$. Each metric is normalized using reference baselines from pre-attack operation:

$$\text{thr}_{\text{norm}} = \frac{\overline{\text{thr}}}{\max(\text{thr}_{\text{ref}}, \epsilon)},$$
$$\text{lat}_{\text{norm}} = \frac{\overline{\text{lat}}}{\max(\text{lat}_{\text{ref}}, \epsilon)},$$
$$\text{loss}_{\text{norm}} = \frac{\overline{\text{loss}}}{\max(\text{loss}_{\text{ref}}, \epsilon)}.$$

We form a composite QoS index

$$\mathcal{Q} = w_1 \,\text{thr}_{\text{norm}} - w_2 \,\text{lat}_{\text{norm}} - w_3 \,\text{loss}_{\text{norm}}, \qquad w_i \geq 0,\ \sum w_i = 1,$$

and apply the early-stop/rollback guard from Sec. VIII-E: if $\mathcal{Q}$ drops by $> \delta$ for U consecutive windows, we pause reconfiguration and restore the last stable ruleset.

b) Security and collaboration metrics.: Security is captured by attack-path coverage COV and collaboration metrics from Sec. VIII-H: union-based gain GAIN and redundancy ratio RED.

c) Complexity metrics.: We report (i) selection wall-time per epoch and (ii) rule-commit count $\mathcal{C}_{\text{rule}}$, upper-bounded by $k + \sum_{h \in \mathcal{H}_k} \deg_S(h)$ and shaped by the token-bucket limiter (Sec. VIII-E).

d) Protocols and reporting.: All methods share seeds, budgets $B \in \{1, 3, 5, 8\}$, and load sweep $\rho \in [0.2, 1.2]$ (Sec. VIII-I). We present tri-objective plots: (a) Security–Complexity scatter (color-coded by $\mathcal{Q}$), (b) $\mathcal{Q}$ vs. ρ with 95% CIs, and (c) bar charts of $\mathcal{C}_{\text{rule}}$ with paired differences. Significance uses Wilcoxon signed-rank with Holm correction; we also report Cliff's δ (Sec. VIII-J).

e) Observations (brief).: Across loads and budgets, NoEC attains higher COV at comparable or lower $\mathcal{C}_{\text{rule}}$ than BAP; vs. TGCESA, NoEC maintains similar $\mathcal{Q}$ while reducing churn due to budget-aware selection. Under high ρ, the load-aware cap $k(\rho)$ preserves $\mathcal{Q}$ with modest security trade-offs (Sec. VII-A).

G. Slice-Aware Evaluation (5G/6G)

We evaluate NoEC in a sliced setting with $\mathcal{L} = \{\text{eMBB, URLLC, mMTC}\}$. Each slice $\ell \in \mathcal{L}$ has hosts H_ℓ, services S_ℓ, budget B_ℓ, and a per-epoch commit cap $C^{(\ell)}_{\max}$; commits are gated by per-slice token buckets while respecting a global TCAM limit $\sum_\ell C^{(\ell)}_{\max} \leq C_{\text{total}}$.

a) Per-slice selection and costs.: Selection is performed per slice (independently) using

$$\Psi_\ell(h) \;=\; \frac{\deg_{S_\ell}(h)}{c'_\ell(h)}, \qquad c'_\ell(h) = c_\ell(h)\,\big(1 + \gamma_\ell\, r_\ell(h)\big),$$

with optional trust/risk shaping inherited from Sec. IV-J. Cross-slice arbitration only applies to TCAM admission when $\sum_\ell C^{(\ell)}_{\max}$ approaches C_{total} (URLLC priority first).

b) QoS guards and URLLC latency thresholds.: For each slice we track a composite QoS index

$$\mathcal{Q}_\ell = w_{1,\ell}\,\overline{\text{thr}}_\ell - w_{2,\ell}\,\overline{\text{lat}}_\ell - w_{3,\ell}\,\overline{\text{loss}}_\ell.$$

URLLC has an explicit latency guard: if $\overline{\text{lat}}_{\text{URLLC}} > \tau_{\text{URLLC}}$ for U consecutive windows, selection on URLLC pauses and reverts to the last stable ruleset; eMBB/mMTC continue under their own guards. Default τ_{URLLC} and window lengths are reported with CIs.

c) Cross-slice isolation metric.: We quantify isolation by the worst normalized spillover from reconfiguration on slice ℓ to any other slice $\ell' \neq \ell$:

$$\text{ISO}_{\ell \to \ell'} \;:=\; \frac{\mathcal{Q}^{\text{after}}_{\ell'} - \mathcal{Q}^{\text{before}}_{\ell'}}{\max\{|\mathcal{Q}^{\text{before}}_{\ell'}|,\, \epsilon\}},$$

and report $\max_{\ell \neq \ell'} \text{ISO}_{\ell \to \ell'}$. We target $|\text{ISO}| \leq \varepsilon_{\text{iso}}$ (e.g., 0.05); violations trigger reduced $C^{(\ell)}_{\max}$ and/or k_ℓ in subsequent epochs.

The URLLC slice evaluation uses a strict latency guard with a default threshold of $\tau_{\text{URLLC}} = 1$ ms; the choice of this value and its 3GPP-based justification are detailed in Appendix B.

d) Massive-host regime.: We scale $|H_\ell|$ and $|S_\ell|$ while keeping per-slice work near-linear: discovery/refresh $\mathcal{O}(|E_\ell|)$, selection $\mathcal{O}(|H_\ell| \log |H_\ell|)$. Global overhead aggregates across slices; TCAM admissions are shaped so instantaneous footprint remains within C_{total}.

H. Coverage and Collaborative Defense Metrics

Let $\mathcal{P}$ denote the set of simple host–server paths under consideration. For a host $h \in H$, we define

$$\mathcal{P}(h) \;:=\; \{\, p \in \mathcal{P} \;:\; p \text{ contains an incidence } (h, s) \in E \,\},$$

that is, $\mathcal{P}(h)$ is the set of paths that traverse at least one edge incident to h. Given a budget-feasible selection $\mathcal{H}_k \subseteq H$, we define the set of detached incidences

$$\mathcal{E}_{\text{det}} \;=\; \{(h, s) \in E :\; h \in \mathcal{H}_k\},$$

and the corresponding union of covered paths

$$\mathcal{U}(\mathcal{H}_k) \;:=\; \bigcup_{h \in \mathcal{H}_k} \mathcal{P}(h) \;=\; \{\, p \in \mathcal{P} :\; p \cap \mathcal{E}_{\text{det}} \neq \emptyset \,\}.$$

Intuitively, $\mathcal{U}(\mathcal{H}_k)$ collects all paths that are "touched" by at least one detached edge in $\mathcal{E}_{\text{det}}$, and coverage metrics (e.g., COV_ℓ) are derived from the size of this set relative to $\mathcal{P}$.

a) Attack-Path Coverage.: The primary metric is the fraction of paths disrupted by the selection:

$$\text{COV}(\mathcal{H}_k) \;:=\; \frac{|\mathcal{U}(\mathcal{H}_k)|}{|\mathcal{P}|} \;\in [0, 1].$$

For single hosts we write $\text{COV}(h) := \text{COV}(\{h\}) = |\mathcal{P}(h)|/|\mathcal{P}|$.

b) Collaborative Coverage Gain.: To separate synergy from mere averaging, we compare the union coverage with the mean single-host coverage of the selected hosts:

$$\text{GAIN}(\mathcal{H}_k) \;:=\; \text{COV}(\mathcal{H}_k) \;-\; \frac{1}{k} \sum_{h \in \mathcal{H}_k} \text{COV}(h).$$

Positive GAIN indicates that the joint selection covers disproportionately more paths than the average of its constituents (synergy), whereas values near zero suggest limited collaboration.

c) Redundancy Ratio.: We also report a normalized overlap indicator:

$$\text{RED}(\mathcal{H}_k) \;:=\; 1 \;-\; \frac{|\mathcal{U}(\mathcal{H}_k)|}{\sum_{h \in \mathcal{H}_k} |\mathcal{P}(h)|} \;\in [0, 1],$$

which is the fraction of per-host covered paths that are *redundant* due to overlaps. Lower RED means the selection covers diverse (less-overlapping) path sets; higher values signal duplication.

d) Reporting.: Unless stated otherwise, we compute COV, GAIN, and RED over R randomized trials and report means with 95% confidence intervals. When path enumeration is large, we approximate $\mathcal{P}$ via bounded-length simple paths or a fixed-seed random walk sampler; the same sampler is reused across methods to ensure fair comparison.

I. Baseline Configuration and Fairness Protocol

To ensure comparability across methods, we standardize the following protocol.

a) Aligned budgets and cadence.: All methods operate under the same per-epoch budget B and shuffle period T_{shuf}. When a method internally selects k hosts, k is constrained by the same budget and (when enabled) the load-aware cap $k(\rho)$ (Sec. VII-A).

b) Shared randomness and traffic.: Unless stated otherwise, we evaluate with the same random seeds and traffic profiles across methods: (i) offered-load sweep $\rho \in \{0.2, 0.4, 0.6, 0.8, 1.0, 1.2\}$; (ii) DDoS profiles (volumetric, state-exhaustion, application-layer); and (iii) targeted-adversary exponents $\alpha \in \{0, 0.5, 1, 1.5, 2\}$.

c) QoS guard and rollback.: We apply an identical composite QoS index $\mathcal{Q} = w_1 \overline{\text{thr}} - w_2 \overline{\text{lat}} - w_3 \overline{\text{loss}}$ and the same early-stop/rollback policy: reconfiguration pauses if $\mathcal{Q}$ drops by $> \delta$ for U consecutive windows (defaults: $\delta = 0.1$, $U = 3$), after which the last stable ruleset is restored (Sec. VIII-E).

d) Hyperparameter tuning.: We use a common grid for each family and report best-validation settings under the fairness guard: budget points $B \in \{1, 3, 5, 8\}$, batch size $B_{\max} \in \{50, 100, 200\}$, batch window $\Delta t \in \{100, 200, 500\}$ ms, and (when applicable) weightings (λ, γ) for capacity/path-aware variants (Sec. IV-I). Tuning uses the same seeds and traffic as testing, with folds partitioned by seeds.

e) Complexity reporting.: We report (i) selection wall-time per epoch and (ii) rule-commit count C_{rule}, both as means with 95% CIs over R runs. When a commit limiter is enabled, we include its token-bucket trace for all methods.

f) Stopping and plotting conventions.: Runs terminate early on QoS rollback or after a fixed horizon. Plots show means (solid) with 95% CIs (shaded); markers denote budget points. Table captions state $(B, T_{\text{shuf}}, R, \text{seeds})$ explicitly.

J. Statistical Analysis: Significance and Effect Sizes

We assess statistical significance and practical relevance for each metric under comparison (e.g., coverage COV, composite QoS $\mathcal{Q}$).

a) Paired setup and aggregation.: All methods are evaluated on the *same* random seeds and traffic profiles. Let $\{(x_r^{\text{A}}, x_r^{\text{B}})\}_{r=1}^{R}$ denote paired outcomes for methods A and B (e.g., NoEC vs. a baseline) under identical runs $r = 1, \ldots, R$. We analyze the paired differences $\Delta_r = x_r^{\text{A}} - x_r^{\text{B}}$.

b) Confidence intervals (paired bootstrap).: We form a 95% CI for the mean paired difference $\overline{\Delta} = \frac{1}{R} \sum_r \Delta_r$ via a paired bootstrap with $B = 10{,}000$ resamples: draw index multisets $\mathcal{I}_b$ of size R with replacement, compute $\overline{\Delta}^{(b)} = \frac{1}{R} \sum_{r \in \mathcal{I}_b} \Delta_r$, and take the $(2.5, 97.5)$ percentiles of $\{\overline{\Delta}^{(b)}\}_{b=1}^{B}$.

c) Significance testing (nonparametric).: We apply the Wilcoxon signed-rank test over $\{\Delta_r\}$ (two-sided) to obtain a p-value without assuming normality. When normality is indicated by Shapiro–Wilk ($p > 0.05$), we may additionally report a paired t-test result for reference.

d) Multiple comparisons.: Across M simultaneous hypotheses (e.g., multiple baselines and budget points), we control family-wise error using Holm–Bonferroni: sort $p_{(1)} \leq \cdots \leq p_{(M)}$ and compare $p_{(i)}$ to $\alpha/(M - i + 1)$ at level $\alpha = 0.05$.

e) Effect size (Cliff's δ).: To complement p-values, we report Cliff's δ for the paired differences:

$$\delta \;=\; \frac{\#\{(r,s) : \Delta_r > \Delta_s\} \;-\; \#\{(r,s) : \Delta_r < \Delta_s\}}{R^2},$$

interpreting $|\delta|$ as: negligible (< 0.147), small (< 0.33), medium (< 0.474), large (≥ 0.474). We also report the median difference $\mathrm{median}(\Delta_r)$.

f) Reporting and plots.: Tables list $\overline{\Delta}$ ($\pm$ 95% CI), p (Holm-adjusted), and δ. Plots show means (solid) with 95% CIs (shaded). Significance is annotated with $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$ (adjusted). Captions state (R, seeds) and the test used.

K. Topologies and Workload Generation

We evaluate on bipartite host–server graphs synthesized with controlled degree structure and on matched workloads. The generation is driven by explicit distributions and fixed random seeds to ensure reproducibility.

a) Bipartite degree specification.: Let $|H|$ and $|S|$ be the numbers of hosts and servers with target mean degrees $\overline{d}_H$ and $\overline{d}_S$ such that $|H|\overline{d}_H = |S|\overline{d}_S = |E|$. We sample host and server degrees from user-chosen families:

$$d_H \sim \mathcal{D}_H \in \{\mathrm{PL}(\zeta_H, d_{\min}, d_{\max}),\ \mathrm{ER}(\lambda_H),\ \mathrm{CP}(\mu, \rho)\},$$
$$d_S \sim \mathcal{D}_S \in \{\mathrm{PL}(\zeta_S),\ \mathrm{ER}(\lambda_S),\ \mathrm{CP}(\cdot)\}.$$

where PL is a truncated power law, ER is a Poisson/Erdős–Rényi-like degree, and CP denotes a core–periphery mix with core fraction ρ and core mean μ. Degrees are adjusted to satisfy $\sum_H d_H = \sum_S d_S$ by a minimal rounding step.

L. Optimal (Oracle) Baseline on Small Instances

To contextualize greedy selection, we compute an optimal baseline on small instances via a 0–1 knapsack over hosts. Let binary variables $x_h \in \{0, 1\}$ indicate whether host $h \in H$ is selected. With per-host cost $c(h)$, budget B, and server-incidence $\deg_S(h)$, the incidence-detachment objective is

$$\max_{x \in \{0,1\}^{|H|}} \quad \sum_{h \in H} \deg_S(h)\, x_h \qquad \text{s.t.} \quad \sum_{h \in H} c(h)\, x_h \;\leq\; B.$$

This is a classical knapsack MILP that we solve exactly on small/medium graphs using an off-the-shelf solver with default settings and a $<1\%$ MIP gap tolerance. For the capacity-/path-aware variant (Sec. IV-I), replace $\deg_S(h)$ with $\widetilde{\deg}_S(h) = \sum_{(h,s) \in E} w_{hs}$.

a) Optimality gap.: Given the oracle value $F^\star$ and the method value F (e.g., NoEC), we report the relative gap

$$\mathrm{GAP} \;:=\; \frac{F^\star - F}{F^\star} \;\in [0, 1].$$

We aggregate GAP over seeds/runs and show means with 95% CIs (Sec. VIII-J). When GAP is near zero, greedy closely tracks the oracle; positive values quantify the room to optimality at that budget.

b) Scope.: Because knapsack is NP-hard, oracle runs are restricted to small/medium instances (e.g., $|H|$ up to a few hundred depending on cost/degree distributions). Larger scenarios use greedy only; where both are feasible, we include the oracle line/marker in the plots and the gap in table captions.

M. NoEC vs. BAP: Coverage and Multi-Host Collaboration

We compare NoEC against BAP under the fairness protocol (Sec. VIII-I) and paired statistical analysis (Sec. VIII-J). Security is quantified by attack-path coverage $\mathrm{COV}(\mathcal{H}_k)$ and collaboration metrics (Sec. VIII-H): union-based *GAIN* and *RED* (redundancy ratio). Unless stated otherwise, we sweep budgets $B \in \{1, 3, 5, 8\}$ and offered loads $\rho \in [0.2, 1.2]$, averaging over R seeds with 95% CIs.

a) Results overview.: Across all B and ρ, NoEC improves COV relative to BAP while exhibiting lower RED, indicating less overlap among selected hosts' covered paths. The collaboration gain GAIN is consistently positive for NoEC (synergy beyond the average single-host coverage), whereas BAP often shows near-zero GAIN due to selecting cheaper but overlapping hosts.

b) Significance and effect sizes.: We report paired CIs for $\Delta = \mathrm{COV}_{\mathrm{NoEC}} - \mathrm{COV}_{\mathrm{BAP}}$ and Wilcoxon signed-rank p-values with Holm correction across budgets. Effect sizes (Cliff's δ) are included alongside median differences (Sec. VIII-J).

IX. Conclusion

This paper introduced NoEC, a novel MTD strategy for DDoS mitigation. The core idea of NoEC is to prioritize the shuffling of hosts with a higher number of connections to critical servers, thereby maximizing defensive effectiveness. Experimental results show that NoEC achieves significantly lower computational complexity than BAP and other MTD approaches, with its complexity largely independent of the attack pattern. This makes NoEC a cost-effective and scalable solution, particularly suited for large-scale networks such as SDNs and next-generation networks (e.g., 5G/6G). For future work, we plan to incorporate ML techniques to identify critical edges and dynamically guide the shuffling strategy

Acknowledgment

This research is partially supported by the European Union's Horizon Europe research and innovation program under the RIGOUROUS project (Grant No. 101095933).

References

[1] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3930–3946, 2023.

[2] A. Javadpour, A. K. Sangaiah, F. Ja'Fari, P. Pinto, H. Memarzadeh-Tehran, S. Rezaei, and F. Saghafi, "Toward a secure industrial wireless body area network focusing mac layer protocols: an analytical review," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 2028–2038, 2022.

[3] Z. Abdelhay, Y. Bello, and A. Refaey, "Toward zero-trust 6gc: A software defined perimeter approach with dynamic moving target defense mechanism," *IEEE Wireless Communications*, vol. 31, no. 2, pp. 74–80, 2024.

[4] A. Javadpour, P. Pinto, F. Ja'fari, and W. Zhang, "Dmaidps: a distributed multi-agent intrusion detection and prevention system for cloud iot environments," *Cluster Computing*, pp. 1–18, 2022.

[5] A. Javadpour, F. Ja'fari, T. Taleb, F. Turkmen, and C. Benzaïd, "Beyond reinforcement learning for network security: A comprehensive survey and tutorial," *Journal of Information Security and Applications*, vol. 96, p. 104294, 2026.

[6] J.-H. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 709–745, 2020.

[7] Y. Cao, K. Liu, Y. Lin, L. Wang, and Y. Xia, "Deep reinforcement learning based self-evolving moving target defense approach against unknown attacks," *IEEE Internet of Things Journal*, 2024.

[8] A. H. Abdi, L. Audah, A. Salh, M. A. Alhartomi, H. Rasheed, S. Ahmed, and A. Tahir, "Security control and data planes of sdn: A comprehensive review of traditional, ai and mtd approaches to security solutions," *IEEE Access*, 2024.

[9] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and C. Benzaïd, "A comprehensive survey on cyber deception techniques to improve honeypot performance," *Computers & Security*, p. 103792, 2024.

[10] A. Javadpour, F. Ja'Fari, T. Taleb, and C. Benzaïd, "A mathematical model for analyzing honeynets and their cyber deception techniques," in *2023 27th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2023, pp. 81–88.

[11] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Enhancing 5g network slicing: Slice isolation via actor-critic reinforcement learning with optimal graph features," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 31–37.

[12] J. Steinberger, B. Kuhnert, C. Dietz, L. Ball, A. Sperotto, H. Baier, A. Pras, and G. Dreo, "Ddos defense using mtd and sdn," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[13] X. Luo, Q. Yan, M. Wang, and W. Huang, "Using mtd and sdn-based honeypots to defend ddos attacks in iot," in *2019 Computing, Communications and IoT Applications (ComComAp)*. IEEE, 2019, pp. 392–395.

[14] A. Aydeger, M. H. Manshaei, M. A. Rahman, and K. Akkaya, "Strategic defense against stealthy link flooding attacks: A signaling game approach," *IEEE Transactions on Network Science and Engineering*, 2021.

[15] Y. Zhou, G. Cheng, S. Jiang, Y. Zhao, and Z. Chen, "Cost-effective moving target defense against ddos attacks using trilateral game and multi-objective markov decision processes," *Computers & Security*, vol. 97, p. 101976, 2020.

[16] J. Narantuya, S. Yoon, H. Lim, J.-H. Cho, D. S. Kim, T. Moore, and F. Nelson, "Sdn-based ip shuffling moving target defense with multiple sdn controllers," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks–Supplemental Volume (DSN-S)*. IEEE, 2019, pp. 15–16.

[17] Z. Liu, Y. He, W. Wang, S. Wang, X. Li, and B. Zhang, "Aeh-mtd: Adaptive moving target defense scheme for sdn," in *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*. IEEE, 2019, pp. 142–147.

[18] A. Chowdhary, A. Alshamrani, D. Huang, and H. Liang, "Mtd analysis and evaluation framework in software defined network (mason)," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2018, pp. 43–48.

[19] Y. Shi, H. Zhang, J. Wang, F. Xiao, J. Huang, D. Zha, H. Hu, F. Yan, and B. Zhao, "Chaos: An sdn-based moving target defense system," *Security and Communication Networks*, vol. 2017, 2017.

[20] S. Debroy, P. Calyam, M. Nguyen, R. L. Neupane, B. Mukherjee, A. K. Eeralla, and K. Salah, "Frequency-minimal utility-maximal moving target defense against ddos in sdn-based systems," *IEEE Transactions on Network and Service Management*, 2020.

[21] M. F. Hyder and M. A. Ismail, "Securing control and data planes from reconnaissance attacks using distributed shadow controllers, reactive and proactive approaches," *IEEE Access*, vol. 9, pp. 21 881–21 894, 2021.

[22] C. Medina-López, L. Casado, V. González-Ruiz, and Y. Qiao, "An sdn approach to detect targeted attacks in p2p fully connected overlays," *International Journal of Information Security*, pp. 1–11, 2020.

[23] S.-Y. Chang, Y. Park, and B. B. A. Babu, "Fast ip hopping randomization to secure hop-by-hop access in sdn," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 308–320, 2018.

[24] A. Chowdhary, S. Pisharody, and D. Huang, "Sdn based scalable mtd solution in cloud network," in *Proceedings*

of the 2016 ACM Workshop on Moving Target Defense, 2016, pp. 27–36.

[25] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and B. Yang, "Scema: An sdn-oriented cost-effective edge-based mtd approach," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 667–682, 2023.

[26] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Attack graph-based moving target defense in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1653–1668, 2020.

[27] J. Jafarian *et al.*, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *HotSDN*, 2012.

[28] H. Okhravi *et al.*, "A survey of moving target defenses," *IEEE Communications Surveys & Tutorials*, 2014.

[29] E. Al-Shaer *et al.*, "Towards intelligent moving target defense using software defined networking," in *SecureComm*, 2013.

[30] Y. Liu *et al.*, "Secure routing with route mutation for preventing repeated attacks in sdn networks," in *IEEE ICC*, 2018.

[31] H. Hu *et al.*, "Lightweight port hopping for defeating scanning attacks," *Computer Networks*, 2015.

[32] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," *Proceedings 2002 IEEE Symposium on Security and Privacy*, 2002.

[33] S. Noel and S. Jajodia, "Measuring the impact of network security using attack graphs," *Journal of Network and Systems Management*, 2004.

[34] S. Hong *et al.*, "Towards dynamic network reconfiguration for moving target defense," in *IEEE CNS*, 2016.

[35] L. Zhang *et al.*, "Mitigating reconnaissance attacks via dynamic network topology," *IEEE Transactions on Information Forensics and Security*, 2019.

[36] A. Javadpour, F. Ja'fari, C. Benzaïd, and T. Taleb, "An optimized reinforcement learning based mtd mutation strategy for securing edge iot against ddos attack," *Journal of Information Security and Applications*, vol. 93, p. 104138, 2025.

Symbol	Meaning
H, S	Sets of hosts and servers
E	Host–server edges in the bipartite graph
$c(h)$	Cost to reconfigure/shuffle host h
$\deg_S(h)$	Server-incidence (degree) of host h
$\Psi(h)$	Cost-normalized score $\deg_S(h)/c(h)$
B	Budget per shuffle epoch
k	Number of selected hosts under budget B
T_{shuf}	Shuffle period (epoch duration)
$b(h)$	Bandwidth/capacity proxy for host h
α	Targeting exponent in the adversary model

Appendix A: Symbols and Notation

Appendix B: Experimental Settings and Reproducibility

a) Evaluation protocol.: Unless specified otherwise, results are averaged over $R = 10$ independent runs with distinct seeds; we report mean and 95% confidence intervals (CI) via normal approximation. Each run comprises: (i) topology discovery and metric refresh; (ii) selection under budget B; (iii) rule staging/commit (Sec. VIII-E); (iv) QoS probing over a fixed horizon. For load-sweeps we vary ρ over a grid; for targeted adversaries we sweep $\alpha \in \{0, 0.5, 1, 1.5, 2\}$; for trust-decay ablations we sweep $\lambda \in \{0.6, 0.8, 0.95\}$ (Sec. IV-J). Reconfiguration is paused if the QoS index $\mathcal{Q}$ drops by more than δ for W consecutive windows (Sec. VIII-E); runs that trigger a pause continue after rollback with the last stable ruleset.

b) Randomization and CI reporting.: We fix a base seed of 42 for figure reproducibility and cycle over $\{41, \ldots, 50\}$ for CI bands unless stated otherwise. For discrete metrics (e.g., selection size k) we additionally report the empirical distribution in the supplement.

c) Early-stop guard.: During reconfiguration, if the composite QoS index $\mathcal{Q}$ dips by $> \delta$ for W consecutive windows, we rollback to the last stable rule set and pause selection until the index recovers, after which selection resumes with a reduced k (Sec. VIII-E).

d) Artifacts.: Configuration files for all parameter grids, seeds, and figure scripts are bundled with the supplementary material; path names and environment variables are documented in a single launcher script.

e) URLLC Latency Threshold.: For URLLC slices, we adopt a default latency guard of **1 ms**. This value is directly based on the 3GPP URLLC service requirements specified in TS 22.261, which define an end-to-end latency target of 1 ms for ultra-reliable low-latency communications. This threshold is also widely used in experimental 5G/6G testbeds and prior studies on slice-oriented anomaly detection and dynamic MTD policies. The 1 ms guard is therefore selected as a realistic and standards-compliant baseline for evaluating slice adaptability and latency-sensitive reactions in our framework.

f) Code Availability (Planned Release).: The main Mininet scripts, NoEC controller implementation, and configuration templates used in our experiments are currently undergoing internal review and packaging. Upon completion, they will be publicly released through the official repository of

Parameter	Default	Explored range / notes
Budget per epoch B	3 (toy), app.-specific in eval	$\{1, 2, 3, 5, 8, 10\}$
Shuffle period T_{shuf}	5 s	$\{1, 2, 5, 10, 20\}$ s
Selection size k	budget-feasible	implicit from B and $c(h)$
Per-host cost $c(h)$	measured/assigned	heterogeneous vs. uniform cases
Score $\Psi(h)$	$\deg_S(h)/c(h)$	cost-/capacity-weighted variants (Sec. VIII-D)
Targeting exponent α	1	$\{0, 0.5, 1, 1.5, 2\}$
Bandwidth proxy $b(h)$	link capacity share	normalized to $[0, 1]$
Trust decay λ	0.8	$\{0.6, 0.8, 0.95\}$
Trust coupling β	0.5	$\{0.2, 0.5, 1.0\}$
QoS weights $\mathbf{w}$	$(0.5, 0.3, 0.2)$	simplex with $w_i \geq 0,\ \sum w_i = 1$
QoS drop thresh. δ	0.1	$\{0.05, 0.1, 0.2\}$
QoS window W	3	$\{2, 3, 5\}$ windows
Commit budget $C_{\max}$	switch-specific	scaled with TCAM capacity
Batch size $B_{\max}$	100 rules	$\{50, 100, 200\}$
Batch window Δt	200 ms	$\{100, 200, 500\}$ ms
Load factor ρ	0.6	grid in $[0.2, 1.2]$
Repetitions R	10	$\{5, 10, 20\}$
Random seed	42 (base)	seeds $\{41, \ldots, 50\}$

TABLE VI
DEFAULT SETTINGS AND EXPLORED RANGES USED ACROSS EXPERIMENTS.

our research lab at http://www.mosaic-lab.org/. The repository will also include a concise README describing software dependencies, topology initialisation, and step-by-step instructions for reproducing the experimental results presented in Section IV.