When ECN Lies: Unfairness and Exploitation in L4S Architectures

Lucas Jaiel de Sousa Correia[1], Alireza Shirmarz[2], Fábio Luciano Verdi[2], Paulo Ditarso Maciel Jr.[1] and Leandro C. de Almeida[1]

[1]Academic Unit of Informatics – Federal Institute of Paraíba (IFPB)
João Pessoa – PB – Brazil

[2]Federal University of São Carlos (UFSCar)
Sorocaba – SP – Brazil

lucas.correia@academico.ifpb.edu.br, {ashirmarz,verdi}@ufscar.br

{paulo.maciel,leandro.almeida}@ifpb.edu.br

Abstract. *Low Latency, Low Loss and Scalable Throughput (L4S) is a recent architecture proposed by the IETF that enables Internet applications to achieve low queuing latency, low congestion loss, and scalable throughput control. The architecture introduces incremental changes to both hosts and network nodes. On the host side, L4S must incorporate a novel variant of a scalable congestion control capable of recognizing the congestion signals. At network nodes, L4S brings a dual-queue coupled mechanism, in which one queue serves Classic traffic while another serves Scalable traffic, enabling fair bandwidth sharing and harmonious coexistence between TCP flavors. Despite its promise, the architecture faces important security challenges. One of them is the so-called unresponsive Explicit Congestion Notification (ECN) attack, in which a malicious (non-L4S) flow exploits the low-latency queue, causing starvation of compliant traffic. In this work, we conduct an in-depth evaluation of the impact of unresponsive ECN attacks in networks with L4S support. Furthermore, we propose and validate a data-plane-based mitigation mechanism, implemented in P4, capable of identifying and penalizing non-compliant flows, thereby restoring fairness and protecting the low-latency service guarantees of L4S.*

1. Introduction

The technological evolution of recent decades has consolidated an era of real-time applications, in which latency and its variation (jitter) are critical factors for both Quality of Service (QoS) and Quality of Experience (QoE). Videoconferencing platforms, interactive streaming, IoT systems for autonomous vehicles, and even remote surgeries demand responses within sub-milliseconds, at the risk of compromising essential functionalities or even endangering lives [Letourneau et al. 2021, Schepper et al. 2023].

These strict requirements expose the limitations of traditional network architectures, which were designed to prioritize throughput over precise delay control. In this context, queueing delay, i.e., the delay caused by packet buffering on routers, is a major contributor. Even in high-capacity networks, the lack of efficient congestion management mechanisms leads to persistent queues, thereby increasing the latency and degrading the QoS/QoE of latency-sensitive applications. The Low Latency, Low Loss

and Scalable Throughput (L4S) architecture, standardized in RFC 9330, is a recent advance that responds to this challenge [Briscoe et al. 2023].

The primary objective of the L4S architecture is to ensure low latency and scalable throughput, even under congested network conditions. The coexistence between scalable (L4S) and classical (e.g., TCP CUBIC) traffic, achieved through a dual-queue coupled mechanism, in which congestion control is linked to ensure throughput fairness, is essential to make this goal feasible in heterogeneous environments such as the Internet. The distinction between these flows is made through the Explicit Congestion Notification (ECN) field in the IP header. L4S flows use code "01" to indicate that they implement smooth and responsive congestion control algorithms, gradually reducing their transmission rate upon detecting congestion markings (CE). In contrast, classic flows use "00" or disregard ECN altogether, relying on more aggressive mechanisms such as abrupt throughput reductions following packet loss [Briscoe et al. 2023].

However, this separation assumes that participants play by the rules, something that does not always reflect reality. The ECN bits, fundamental for traffic separation, are located in the IP header, a layer that is easily accessible and configurable. In this sense, a malicious agent could modify this field, tricking the system into treating regular traffic as prioritized, without adhering to the architecture [Letourneau et al. 2023].

By marking non-compliant packets as ECT(01), an attacker gains privileged access to the low-latency queue, compromising the performance of legitimate applications. Furthermore, fairness in resource distribution is undermined, as malicious flows could consume bandwidth without reducing their transmission rate in response to congestion [Letourneau et al. 2023]. This scenario is not limited to isolated failures: in large and heterogeneous networks such as the Internet, the absence of robust mechanisms to validate flow compliance could render L4S unfeasible.

In this context, the first question that we want to ask is: *What happens when a malicious flow, disguised as L4S, floods the priority queue?*

In the following sections, we will demonstrate how manipulation attacks on the ECN field, a core mechanism for distinguishing between Scalable and Classic flows, can significantly degrade the L4S architecture. By allowing malicious agents to tamper with congestion markers, not only is traffic prioritization compromised, but so is the very logic of coexistence that underpins the promise of low latency and high efficiency.

After detailing the problem, we address the second question: *Is it possible to identify and punish a malicious flow that is consuming priority queue resources?*

In this work, we present a line-rate, data-plane-based defense mechanism to detect and mitigate malicious L4S traffic that abuses priority queues. Our solution is implemented entirely in P4 and is evaluated on a programmable software switch. The proposed approach continuously monitors congestion responsiveness to identify non-compliant flows and enforces penalties directly in the forwarding path. Moreover, our mechanism preserves L4S coexistence guaranties without relying on end-host trust or control-plane intervention, thereby strengthening the practical deployability of L4S in adversarial and heterogeneous network environments.

The contributions of this work are threefold: i) An in-depth quantitative analysis

of the impact of unresponsive ECN manipulation attacks, demonstrating how malicious flows disguised as L4S can degrade latency, fairness, and coexistence guaranties; ii) Design of a data-plane-based detection and mitigation mechanism that identifies non-compliant L4S flows by monitoring congestion responsiveness and enforces penalties directly on the forwarding path; iii) A practical proof-of-concept implementation in P4, showing the feasibility of deploying the proposed defense without end-host modifications.

The remainder of the paper is structured as follows. Section 2 presents the key concepts for a better understanding of L4S architecture. The related work is described in Section 3. An overview of the threat model is presented in Section 4. Section 5 presents our solution. Finally, conclusions are drawn in Section 6.

2. Background

The L4S architecture [Briscoe et al. 2023] consists of three main components:

- A scalable congestion control algorithm at the sender, which reacts promptly and smoothly to congestion signals. Some of the already available protocols being L4S-capable include Prague for TCP and Prague for QUIC [Schepper et al. 2024];
- An Active Queue Management (AQM) mechanism at the network nodes that provides immediate ECN marking as a congestion signal, named DualPI2;
- A modified ECN protocol to distinguish L4S packets from "classic" ones.

The TCP Prague is a scalable congestion control algorithm designed specifically to meet the requirements of the L4S architecture. It is derived from Data Center TCP (DCTCP) [Schepper et al. 2024] and introduces essential safety features to ensure safe deployment over the public Internet [Schepper and Briscoe 2023]. Prague adjusts the congestion window in response to ECN signals in a fine-grained manner, maintaining a stable rate and sub-millisecond queueing delays. Unlike classic controls, its responsiveness does not degrade as the flow rate increases, making it highly scalable, enabling rate control through a scalable signaling mechanism, effective from near-zero transmission rates up to theoretically unlimited bandwidth [Schepper et al. 2024].

Classic ECN only signals the presence of congestion and does not convey its intensity. In contrast, TCP Prague and related schemes (e.g., DCTCP) require information about multiple congestion events within an RTT. To this end, Accurate ECN, as adopted by L4S, uses the 3-bit ACE field in the TCP header to encode the number of CE-marked packets. Although richer feedback mechanisms exist (e.g., byte-level congestion signals), ACE-based verification is more robust, as byte-count feedback can be altered or suppressed by middleboxes [Briscoe et al. 2025].

To ensure coexistence between scalable and classic traffic, the L4S architecture introduces the Dual-Queue Coupled AQM, a mechanism that deploys two queues: i) an L4S queue with a very shallow marking threshold for ECN-based congestion signals; ii) a classic queue managed by a traditional AQM [Schepper et al. 2023]. The innovation of this mechanism lies in its coupling: the drop/mark probability from the classic queue influences the L4S queue, ensuring safe coexistence across traffic classes while preserving low latency for L4S flows. This coupling achieves both latency isolation and bandwidth pooling.

Figure 1 illustrates the overall structure of the Dual-Queue Coupled AQM. An initial classifier distinguishes the incoming packets and directs them to the appropriate queue. Each queue is managed by its own AQM, which computes congestion marking or dropping probabilities. The Classic queue uses a two-stage AQM. The L4S queue has a native AQM that calculates its base marking probability based on an instantaneous delay. To maintain fairness and responsiveness, its final marking probability is defined as the probability derived from the Classic queue. Although the L4S queue is given conditional priority by the scheduler, the coupling ensures that Classic flows can still exert influence. When the Classic queue builds up, L4S receives more signals and reduces its rate, preserving fairness and low latency.

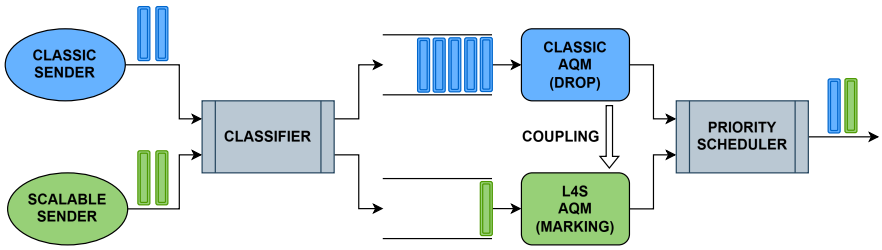


Figure 1. DualQ Coupled AQM schematic (adapted from RFC 9332).

L4S introduces a modified ECN mechanism that employs a more immediate and frequent marking approach than packet drops, while the transport layer responds to each mark with a reduced and smoother adjustment. The significantly more frequent ECN signals, combined with finer-grained transport responses, enable very low queueing delays without compromising link utilization. This low delay can be sustained even under high network load [Schepper and Briscoe 2023].

The ECN field in the IP header consists of two bits, allowing four possible codepoints. Before enabling ECN, the protocol verifies whether the receiver is capable of processing ECN feedback. The connection is not considered ECN-capable (non-ECT) if either endpoint does not support ECN (default behavior of legacy hosts). To indicate L4S-capable traffic, flows are required to explicitly identify themselves by marking packets with the repurposed ECT(1) codepoint (ECN value ‘01’) in the IP header.

A fundamental premise of the L4S architecture is that participating flows are responsive to congestion feedback. In other words, it relies on the assumption that flows classified as L4S will react appropriately and quickly to congestion signals, particularly those delivered via ECN marks [Schepper and Briscoe 2023]. Without such responsiveness, L4S flows could monopolize bandwidth or cause instability within the network. This assumption of flow responsiveness is not merely a design choice.

Although ECN and AQM constitute the technical foundation of the L4S architecture, they also expose new attack surfaces that can be exploited by malicious actors and misconfigured endpoints. These vulnerabilities, referring to them as “undesirable flows”, are categorized into three distinct groups: *misbehaving flows*, *unresponsive flows*, and *malformed flows* [Letourneau et al. 2023]. Misbehaving flows are defined as those that deliberately or unintentionally violate expected protocol behavior. Unresponsive flows are characterized by their failure to react to

standard congestion signals, such as ECN marks, packet drops, or increased latency. Such flows may be legitimate (e.g., certain UDP or VoIP traffic) or maliciously engineered to disregard feedback while feigning compliance. Finally, malformed flows refer to traffic patterns that, despite potentially legitimate origins, adversely affect low-latency service performance due to structural irregularities, most commonly burstiness [Letourneau et al. 2023].

In this work, our attention is focused on the behavior of unresponsive flows. In that case, we will examine how such a flow can disrupt the L4S architecture.

3. Related Work

The emergence of low-latency services has boosted the adoption of the L4S (Low Latency, Low Loss, and Scalable Throughput) architecture as a promising approach for the coexistence of classic traffic and latency-sensitive flows. However, security and robustness under adversarial behavior have become critical concerns.

3.1. Characterizing Attacks and Vulnerabilities in L4S

The impact of malicious flows in L4S environments was initially explored by [Letourneau et al. 2021], in which the *unresponsive ECN* attack was identified. In this setting, an attacker manipulates the behavior of the protocol to ignore ECN congestion signals, leading to unfair bandwidth acquisition and severe latency degradation for legitimate traffic. Subsequent work expanded this analysis by showing that even a small fraction of unresponsive flows, yet compatible with ECN, can subvert the intended behavior of DualQ Coupled AQM and compromise the L4S coexistence guarantees [Letourneau et al. 2023].

3.2. Parameter Manipulation and Selfish Behavior in QUIC

The ease of manipulating user-space protocols such as QUIC introduces new attack vectors. The authors in [Joarder et al. 2025] demonstrated that selfish clients can perform manipulation of congestion control parameters, adjusting parameters (e.g., LRF and CAGR) in user space to gain unfair resource advantages, often manifesting as flows that effectively become unresponsive to network signals. While that work emphasizes host-side exploitation, our approach targets a data-plane defense that can identify such behavior directly within the switch, without relying on end-host compliance or trust.

3.3. Machine-Learning-Based Detection of Low-Latency Attacks

To mitigate threats against low-latency traffic, the work in [Cogranne et al. 2025a] proposed hybrid deep-learning models that combine autoencoders and transformers to detect anomalies in multivariate time series and identify unresponsive ECN attacks. Although such models can achieve high detection accuracy with low false-positive rates, they typically operate outside the data plane and/or require telemetry collection (e.g., INT) for offline or delayed processing. In contrast, our solution implements both detection and mitigation at line rate entirely in P4, avoiding the reaction delay inherent to control-plane intervention or external statistical/ML pipelines. Additionally, the authors presented in [Cogranne et al. 2025b] a lightweight autoadaptive traffic model (PCA-based, unsupervised) that isolates the "attack footprint" by rejecting legitimate traffic as nuisance variation; our work differs by emphasizing in-switch enforcement rather than purely analytical or offline identification.

3.4. In-Data-Plane Feedback and Queue Management

Recent frameworks such as Network-Assisted Congestion Feedback (NCF) leverage programmable data planes (P4) to isolate flows (e.g., mice vs. elephants) and generate richer, sub-RTT congestion feedback to senders, improving performance and fairness in cooperative traffic settings [Fathalli et al. 2026]. While our approach also relies on P4-programmable switches, our focus is defensive and adversarial: instead of optimizing cooperative traffic, we actively monitor congestion responsiveness and apply penalties to non-conformant flows, thereby strengthening the practical deployability of L4S in heterogeneous and potentially hostile environments.

3.5. Comparative Summary

Our work complements these studies by providing a deeper quantitative analysis of such adversarial impacts, focusing on how ECN manipulation exposes the fragility of low-latency queues under hostile conditions. Table 1 compares our work with related studies by focus, enabling technologies, deployment location, and attack response.

Table 1. Comparison of the proposed approach with related work.

Work	**Main focus**	**Technology**	**Logic placement**	**Response to attack**
[Letourneau et al. 2021] [Letourneau et al. 2023]	Threat characterization	L4S testbed	End-host/ analysis	N/A (study)
[Cogranne et al. 2025a] [Cogranne et al. 2025b]	Detection/ modeling	P4 (telemetry) + ML	External (offline)	Statistical/ ML detection
[Joarder et al. 2025]	QUIC-based attack	QUIC software	End-hosts	N/A (attack)
[Fathalli et al. 2026]	Feedback/ performance	P4	Data plane	Rich feedback (NCF)
This work	Defense and mitigation	P4	Data plane	Real-time penalties

4. Unresponsive ECN Attack

The L4S architecture faces critical challenges when malicious clients exploit the lack of flow compliance verification [Letourneau et al. 2021]. In this context, malicious agents can manipulate the behavior of the protocol and bypass the separation logic originally designed to prioritize responsive traffic.

4.1. Attack Description

The attack evaluated in this work relates to the manipulation of the protocol and the generation of unresponsive ECN flows, which occurs when classic congestion control algorithms (such as TCP CUBIC) are manipulated to mark their packets with the ECT(1) codepoint in the ECN field, thus bypassing the logic of traffic separation of the DualPI2 AQM. By tampering with the IP header, these flows deceive the router, which forwards them to the L4S queue, originally designed to prioritize traffic that responds to ECN.

However, unlike legitimate L4S flows (such as TCP Prague), which immediately adjust their sending rate upon detecting CE marks, classic flows retain their original behavior: they ignore congestion signals and only react to packet loss or retransmission timeouts.

This mismatch creates an anomalous asymmetry: while legitimate flows reduce their sending rate in response to CE marks, malicious flows disguised as L4S continue transmitting at high rates. In a scenario with multiple competing flows, the attacker can progressively occupies the L4S queue, monopolizing prioritized resources. The queue, which was originally short and stable, begins to accumulate unresponsive packets, increasing queuing delay and monopolizing bandwidth, thereby degrading and rendering the remaining flows unusable. This situation can be observed in Figure 2, in which malicious packets take advantage of the L4S queue to gain advantages over other flows. We replicated this attack in a controlled environment to understand the adverse effects that a malicious flow can have on the network. The next section describes this lab.

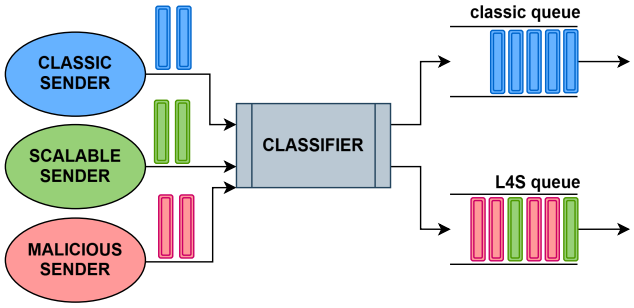


Figure 2. DualQ-AQM attack schematic. The malicious sender takes advantage of the L4S queue, consuming bandwidth resources intended for low-latency flows.

4.2. Attack Evaluation

We adopted a testbed based on the Infrastructure as Code (IaC) approach, composed of six virtual machines (VM), provided with Vagrant and VirtualBox[1]. As shown in Figure 3, the setup is composed of three senders, two receivers, and one router.

At network endpoints, we define three types of senders: *i)* a legitimate L4S sender, which uses the TCP Prague (Linux kernel 5.15.72-prague) congestion control algorithm to represent compliant, low-latency-responsive behavior; *ii)* a legitimate classic sender, running a traditional TCP flow (CUBIC) that adheres to the L4S coexistence guidelines and does not manipulate ECN header fields; and *iii)* a malicious classic sender, which is a traditional TCP flow that falsely claims L4S compatibility by forging the ECT(1) codepoint in the ECN field through iptables-based header rewriting. Both classic senders operate on the Linux kernel version 5.4.0-215-generic.

At the core of the network, we deploy a central router responsible for packet forwarding and congestion management. This router implements the iRED AQM mechanism [De Almeida et al. 2026], configured with a dual-queue coupled design and a target queuing delay of 20ms. The router is instantiated using the BMv2 software switch. To enforce a controlled bottleneck, we rate-limit the egress interface toward the receivers

[1]Artifacts available in the public repository: https://github.com/LucasJaiel/L4S$_S EC$.

to approximately $10\mathrm{Mbit/s}$, since this limit is enforced on a packet-count basis rather than a strict bit-rate basis, the observed throughput may slightly exceed 10 Mbit/s. On the destination side, traffic is directed to two receivers, one handling L4S flows and the other serving classic traffic

We deploy three concurrent flows, each consisting of a single TCP connection generated using Iperf and sustained for 180 seconds. Two flows use classic TCP congestion-control algorithms, one compliant and one malicious, while the third employs TCP Prague (L4S). At the end hosts, we measure throughput in Mbps as the primary performance metric. At the router, we monitor the queueing delay per-queue in milliseconds and collect packet drop statistics from the classic queue and ECN marking statistics from the L4S queue.

4.3. Preliminary Analysis

In this section, we evaluate the impact of a malicious sender on the L4S architecture (described in Figure 3) by quantifying its effects on bandwidth allocation and queuing delay. As described earlier, the malicious sender corresponds to a client running a classic TCP congestion-control algorithm, such as CUBIC, that exploits the low-latency queue by misleading the L4S classifier at the router. Figure 4 compares bandwidth sharing among flows under baseline conditions and in the presence of an ECN-unresponsive attack.

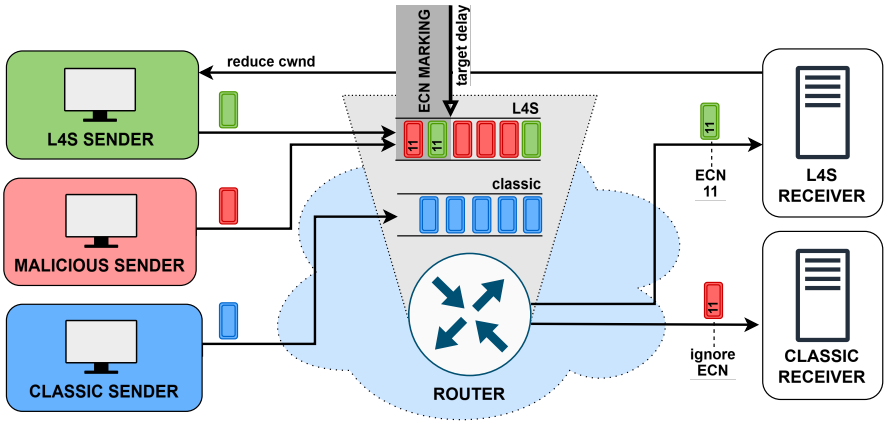
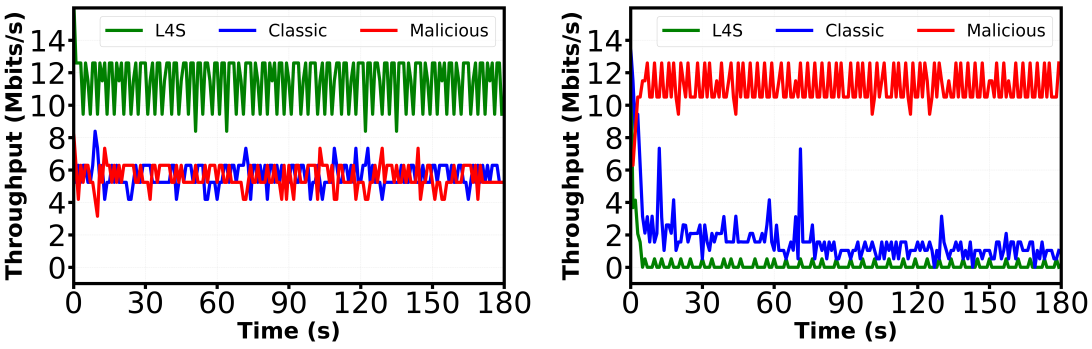


Figure 3. The malicious host sends a stream of packets to the classic receiver, benefiting from the low-latency architecture provided by L4S. The attacker ignores ECN markings and does not reduce the segment sending rate.

Comparing the baseline scenario (Figure 4(a)), without malicious traffic, and the attack scenario (Figure 4(b)), the degradation is evident. In the baseline, bandwidth is distributed among flows, with both L4S and classic flows achieving fair throughput levels, reflecting the expected behavior of the dual-queue coupled mechanism that ensures fairness. However, under attack, malicious flow maintains an aggressive transmission rate of 11.1 Mbps throughout the entire 180-second experiment, while legitimate L4S flow is reduced to a mere 243 Kbps, a reduction of approximately 98% compared to its baseline performance. The legitimate classic flow suffers similarly, achieving only 1.71 Mbps. This stark contrast demonstrates how a single unresponsive flow can completely subvert the fairness guarantees that are fundamental to the L4S architecture.



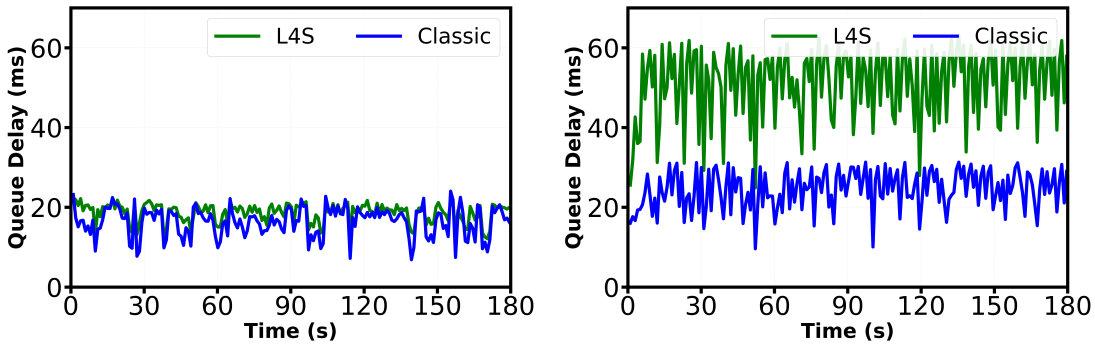
(a) Baseline (b) ECN unresponsive attack

Figure 4. Throughput presented at the egress router bottleneck interface in both the Baseline and the malicious user scenarios and the imbalance caused by the attacker. Malicious sender ignores congestion feedbacks, being supressed only by packet drops caused by bandwidth capacity exhaustion.

As we can also observe from the figure, the malicious sender creates a significant imbalance in the distribution of available bandwidth, consuming much more resources than competing flows. We verified that the attacker maintains a high transmission rate consistently, while the other flows fluctuate between 0.5 and 2.5 Mbps. This occurs because the malicious flow is placed in the low-latency queue (which marks packets with ECN), but it does not respond to the markings generated by the iRED AQM. In contrast, legitimate flows comply with congestion signals and reduce their transmission rates accordingly. This imbalance allows the malicious flow to monopolize most of the available bandwidth, effectively starving the legitimate flows.

One of the primary objectives of the L4S architecture is to achieve extremely low latencies while maintaining high use of the available bandwidth. To achieve this, it employs a coupled dual-queue mechanism designed to prioritize low-latency traffic. However, this characteristic was exploited by a malicious flow that leverages the low-latency queue. Figure 5 presents the quantitative results of the queue delay in both baseline and malicious scenarios. In this case, the green curve in Figure 5(b) represents the L4S queueing delay, which is shared (and thus consumed) by both malicious traffic and legitimate L4S traffic. As we can observe, the delay in the low-latency queue was higher than in the classic queue under attack.

It is important to clarify that this behavior is not expected. Senders using the low-latency queue are supposed to respond to ECN signals by reducing the number of segments they send. However, in this case, the malicious sender ignores these ECN signals and does not react to network congestion, resulting in a bottleneck in the low-latency queue. The measurements reveal that the delay in the L4S queue averaged approximately 50 ms during the attack, with peaks exceeding 60 ms, significantly higher than the configured target delay of 20 ms. This represents a delay increase of 150-200% compared to the baseline scenario, where the L4S queue maintained delays within the expected threshold range of 10-20 ms as designed. The sustained 50+ ms queueing delays observed in our attack scenario would degrade the performance of low-latency applications, potentially causing video freezing, audio desynchronization, or unacceptable input lag. Furthermore, for emerging ultra-low-latency applications such



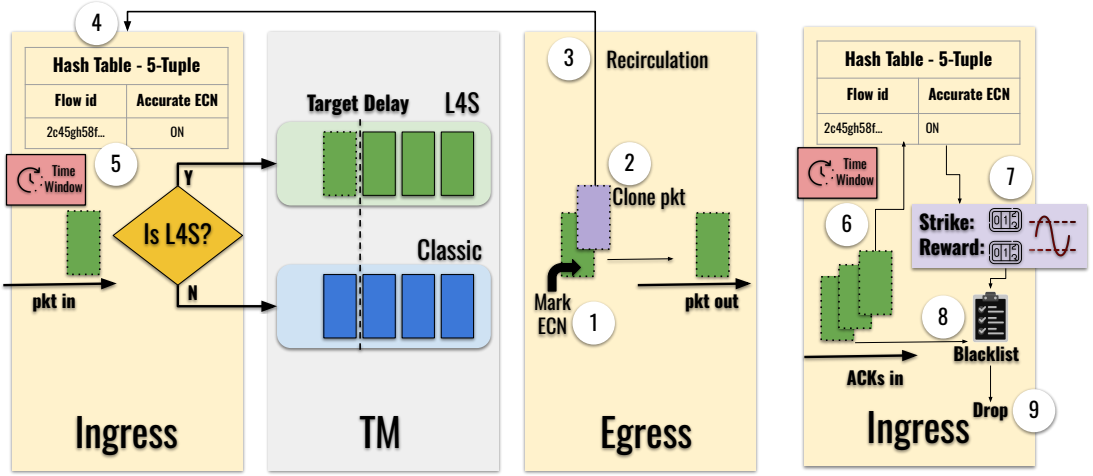
(a) Baseline (b) ECN Unresponsive attack

Figure 5. Queue delay analysis in the evaluated scenarios. The baseline behavior reflects the target threshold imposed by the iRED AQM. In contrast, in the unresponsive attack scenario, the L4S queue is severely impacted, maintaining latency levels that significantly deviate from the standard operational range.

as remote surgery or autonomous vehicle coordination, where even tens of milliseconds can be critical, such queue delays could render these services completely unusable.

5. In-Network ACE-Enabled Malicious Attack Detector

Since the root of the problem lies in the naive flow classification process, we conjecture that an additional mechanism is needed to address this limitation. To this end, we propose a solution that incorporates an auxiliary feature for the flow classification mechanism. Our mechanism, illustrated in Figure 6, operates within the BMv2 switch across three blocks: ingress pipeline, traffic manager, and egress pipeline.



(a) Detection of L4S packets marked with ECN (CE) and generation of notifications to hash table such flows for monitoring.

(b) Mitigation relies on checking if the ACE flag remains responsive in flows previously marked.

Figure 6. Packet processing flow between ingress and egress pipelines, highlighting the recirculation for flow registration and the ACE-based compliance verification.

The verification process begins when congestion is detected on L4S packets in

the egress pipeline (step 1). Upon detecting a CE marking (step 2), the packet is cloned and recirculated back to the ingress pipeline for flow registration (step 3). In the ingress pipeline, recirculated L4S (CE) packets trigger the flow registration process (step 4).

The system extracts the 5-tuple (source IP, destination IP, source port, destination port, and protocol) and applies a CRC32 hash function to map the flow to a specific index position (ranging from 0 to 255) in an indexed memory structure. At this index position, a flag (bit set to 1) is activated to indicate that the flow has experienced congestion and must be monitored for ACE (Accurate ECN) compliance (step 5). This registration also activates a monitoring window to initiate the verification process.

Within this window, incoming ACK packets are intercepted and analyzed (step 6). For each registered flow, the mechanism implements a strike/reward system based on ACE field verification (step 7). The receiver uses the ACE field [Briscoe et al. 2023] to report the accurate number of marked lines seen (within each RTT). When ACK packets arrive, the system examines the ACE counter to determine whether it reflects the congestion signaled previously. Flows that report non-zero ACE values, indicating responsiveness to CE marks, receive rewards in the form of continued access to the low-latency queue. In contrast, flows that report zero ACE values despite receiving CE marks accumulate strikes, signaling unresponsive or malicious behavior.

After exceeding a configurable strike threshold, the flow is classified as malicious and added to a blacklist (step 8). Subsequent packets from blacklisted flows are dropped in the ingress pipeline (step 9), preventing them from exploiting the low-latency queue. This approach provides a dynamic feedback-based authentication mechanism that operates transparently to legitimate flows while effectively neutralizing the ECN unresponsive attack described in Section 3.

5.1. Evaluation & Result Analysis

The analysis of bandwidth and queueing delay presented previously is further supported by the data presented in Figure 7. From these graphs, we can distinguish the three experimental cases: Scenario I (Baseline), Scenario II (Malicious Attack), and Scenario III (Proposal Solution), which illustrate the number of ECN markings. Given the target delay of $20ms$, we observe that during the attack, the delay in the queue consistently exceeds the threshold. Consequently, the AQM is forced to generate a large volume of ECN markings in an attempt to signal congestion to the unresponsive sender. Moreover, results show that Scenario III (proposal solution) consistently yields fewer ECN Marks than II, indicating a more stable queue occupancy and pressure even under attack.

Figure 8 shows the results from an experimental evaluation for Scenario III. With the mitigation enabled, the malicious flow (red) is rapidly pushed to essentially 0 Mbps, while the legitimate L4S (green) and Classic (blue) flows stabilize and share the bandwidth fairly at roughly 11–12 Mbps each, indicating restored fairness in capacity allocation; simultaneously, the queueing delay, after a brief initial transient, remains bounded and relatively stable (mostly around 15–22 ms for both L4S and Classic, with modest jitter), showing that the mitigation also normalizes the L4S queue delay and prevents persistent queue inflation. Again, the green curve in Figure 8(b) represents the delay in L4S queueing, which is shared by malicious traffic and legitimate L4S traffic.

To evaluate the proposed mechanism under realistic application conditions, we

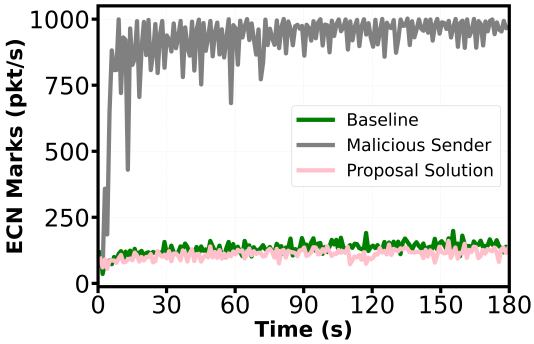
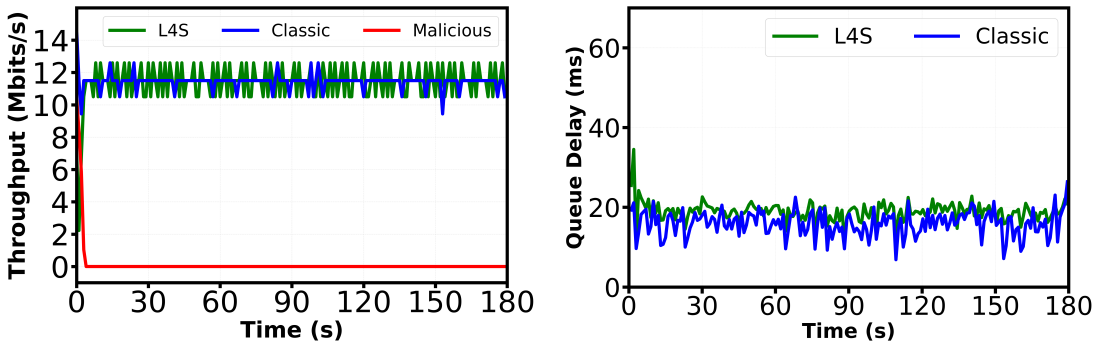


Figure 7. ECN marks by iRED in all evaluated scenarios.



(a) Restoration of fairness in bandwidth distribution.

(b) Normalization of the L4S queue delay.

Figure 8. Experimental evaluation results for Scenario III with the mitigation mechanism implemented.

conducted a final experiment using a real video streaming service. In this setup, the server generated and transmitted a video stream using FFmpeg, while the client also relied on FFmpeg to receive and render the stream, allowing us to emulate a practical end-to-end video delivery pipeline. This experiment extends the network-level analysis by evaluating end-user QoE when mitigation is enabled.

Figure 9 highlights the impact of the proposed data-plane mitigation on the QoE of a video streaming service. When the ECN-unresponsive attack is present (Without Proposal), the FPS distribution collapses toward zero, revealing severe and sustained playback degradation, characterized by freezes and the inability of the player to maintain a usable rendering rate. In contrast, enabling the proposed mechanism restores a stable FPS distribution, with most samples concentrated between approximately 25 and 40 FPS and a median above 30 FPS. By preventing malicious flows from monopolizing the low-latency queue, the proposed approach preserves the performance guarantees required by latency-sensitive video streaming applications.

6. Conclusion & Future Works

This work examined the susceptibility of the L4S architecture to malicious manipulation of ECN signaling by classic TCP flows and introduced a verification mechanism to counteract this vulnerability. Our results demonstrate that ECN forgery by malicious senders compromises the core objectives of L4S. In particular, classic TCP flows that

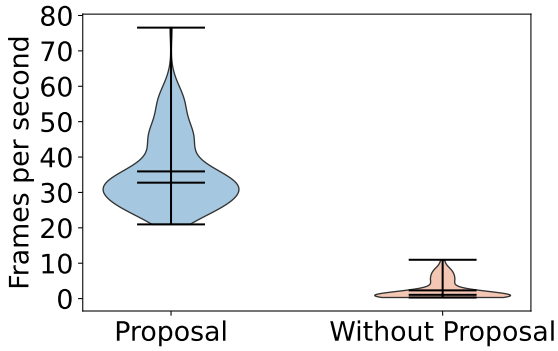


Figure 9. Distribution of video playback FPS at the client (0–160 s). Without the proposal, the FPS collapses toward zero under the ECN-unresponsive attack, while the proposed mechanism sustains a higher and more stable FPS distribution, indicating improved video quality of experience.

falsely signal L4S compliance can dominate bandwidth allocation and induce excessive queueing delay in the low-latency queue, negating the benefits promised by the L4S.

In general, the proposed solution provides a practical and deployable defense against ECN manipulation attacks without requiring modifications to the end hosts or the introduction of new packet header fields. However, it is currently limited to TCP-based L4S traffic, since the verification mechanism relies on the ACE field to assess congestion responsiveness. Moreover, it also assumes the path symmetry between forward and return traffic, as the verification process depends on observing ACK packets on the reverse path. In asymmetric routing scenarios, where ACKs do not traverse the same monitoring point, the mechanism may fail to complete the verification process and thus become ineffective. Despite these limitations, the solution is well-suited for deployment at organizational edge routers, where traffic paths are typically symmetric and under administrative control. In such settings, the mechanism can protect low-latency services by enforcing ECN responsiveness at line rate and preventing malicious flows from exploiting the L4S queue.

Future work should test the mechanism in more diverse and realistic network conditions, analyze how it holds up against smarter adversarial strategies, and measure practical deployment costs and interactions with existing network controls, including validation in larger testbeds and extension to future L4S/ACE-based transports. Also, other protocols such as Prague for QUIC and SCReAM should be evaluated, taking into account that such protocols do not support ACE, and other solutions must be investigated.

Acknowledgments

This work was partially supported by CNPq under grant no. 404509/2025-8 and IFPB.

References

Briscoe, B., Kühlewind, M., and Scheffenegger, R. (2025). More Accurate Explicit Congestion Notification (AccECN) Feedback in TCP. Internet-Draft draft-ietf-tcpm-accurate-ecn-34, Internet Engineering Task Force. Work in Progress.

Briscoe, B., Schepper, K. D., Bagnulo, M., and White, G. (2023). Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture. RFC 9330.

Cogranne, R., Letourneau, M., and Doyen, G. (2025a). A Hybrid Autoencoder–Transformer Model for Detection of Attacks on Low Latency Services. In *2025 International Conference on Advanced Machine Learning and Data Science*.

Cogranne, R., Letourneau, M., Doyen, G., and Nguyen, H. N. (2025b). A Simple yet Accurate Autoadaptive Model of Network Traffic for Detection of Attacks on Low Latency Services. In Ito, A., editor, *Proceedings of the 10th International Conference on Multimedia Systems and Signal Processing (ICMSSP 2025)*, volume 1637 of *Lecture Notes in Networks and Systems*, pages 43–58. Springer.

De Almeida, L. C., Maciel Jr., P. D., Pasquini, R., Papagianni, C., and Verdi, F. L. (2026). iRED: A disaggregated P4-AQM fully implemented in programmable data plane hardware. *IEEE Transactions on Networking*, Early Access.

Fathalli, S., Weyulu, E. N., Zeynali, D., Chandrasekaran, B., and Feldmann, A. (2026). Network-Assisted Congestion Feedback. *IEEE Transactions on Network and Service Management*, 23:1797–1815.

Joarder, Y. A., Sinha, S., Doyen, G., and Fung, C. J. (2025). Exploiting Congestion Control Parameter Manipulation in QUIC for Security Implications. In Cerroni, W., Tortonesi, M., Borsatti, D., Schaeffer-Filho, A. E., Tuncer, D., François, J., and Husák, M., editors, *21st International Conference on Network and Service Management, CNSM 2025, Bologna, Italy, October 27-31, 2025*, pages 1–9. IEEE.

Letourneau, M., Doyen, G., Cogranne, R., and Mathieu, B. (2023). A Comprehensive Characterization of Threats Targeting Low-Latency Services: The Case of L4S. *Journal of Network and Systems Management*, 31(1):19.

Letourneau, M., N'Djore, K. B., Doyen, G., Mathieu, B., Cogranne, R., and Nguyen, H. N. (2021). Assessing the Threats Targeting Low Latency Traffic: the Case of L4S. In *2021 17th International Conference on Network and Service Management (CNSM)*, pages 544–550.

Schepper, K. D. and Briscoe, B. (2023). The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S). RFC 9331.

Schepper, K. D., Briscoe, B., and White, G. (2023). Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S). RFC 9332.

Schepper, K. D., Tilmans, O., Briscoe, B., and Goel, V. (2024). Prague Congestion Control. Internet-Draft draft-briscoe-iccrg-prague-congestion-control-04, Internet Engineering Task Force. Work in Progress.